BIOFEEDFORWARD by Bill Etra

COMPUTER WRESTLING: THE PROGRAM OF CHAMPIONS

XEROXES & OTHER HARD COPY OFF YOUR CRT

A PAYROLL PROGRAM FOR YOUR SMALL BUSINESS

ROM

COMPUTER APPLICATIONS FOR LIVING

CHARGED COUPLES

Volume I, Number 3
September 1977/$2.00

SWTPC announces first dual minifloppy kit under $1,000

SWTPC MINIFLOPPY DISK SYSTEM

Now SWTPC offers complete best-buy computer system with $995 dual minifloppy, $500 video terminal/monitor, $395 4K computer.

SWTPC MINIFLOPPY DISK SYSTEM

$995 MF-68 Dual Minifloppy
You need dual drives to get full benefits from a minifloppy. So we waited to offer a floppy until we could give you a dependable dual system at the right price.

The MF-68 is a complete top-quality minifloppy for your SWTPC Computer. The kit has controller, chassis, cover, power supply, cables, assembly instructions, two highly reliable Shugart drives, and a diskette with the Floppy Disk Operating System (FDOS) and disk BASIC. (A floppy is no better than its operating system, and the MF-68 has one of the best available.) An optional $850 MF-6X kit expands the system to four drives.

$500 Terminal/Monitor
The CT-64 terminal kit offers these premium features: 64-character lines, upper/lower case letters, switchable control character printing, word highlighting, full cursor control, 110-1200 Baud serial interface, and many others. Separately the CT-64 is $325, the 12 MHz CT-VM monitor $175.

SWTPC 6800 COMPUTER SYSTEM

$395 4K 6800 Computer
The SWTPC 6800 comes complete with 4K memory, serial interface, power supply, chassis, famous Motorola MIKBUG® mini-operating system in read-only memory (ROM), and the most complete documentation with any computer kit. Our growing software library includes 4K and 8K BASIC (cassettes $4.95 and $9.95; paper tape $10.00 and $20.00). Extra memory, $100/4K or $250/8K.

Other SWTPC peripherals include $250 PR-40 Alphanumeric Line Printer (40 characters/line, 5 x 7 dot matrix, 75 line/minute speed, compatible with our 6800 computer and MITS/IMSAI); $79.50 AC-30 Cassette Interface System (writes/reads Kansas City standard tapes, controls two recorders, usable with other computers); and other peripherals now and to come.

Enclosed is:

_____ $1,990 for the full system shown above (MF-68 Minifloppy, CT-64 Terminal with CT-VM Monitor).
_____ $995 for the Dual Minifloppy
_____ $325 for the CT-64 Terminal
_____ $175 for the CT-VM Monitor
_____ $395 for the 4K 6800 Computer
_____ $250 for the PR-40 Line Printer
_____ $79.50 for AC-30 Cassette Interface
_____ Additional 4K memory boards at $100
_____ Additional 8K memory boards at $250
_____ Or BAC # __________ Exp. Date _____
_____ Or MC # __________ Exp. Date _____
Name __________ Address __________
City __________ State _____ Zip _____

Southwest Technical Products Corp.
219 W. Rhapsody, San Antonio, Texas 78216
London: Southwest Technical Products Co., Ltd.
Tokyo: Southwest Technical Products Corp./Japan

Introducing Our Disk System.
System 8813
PolyMorphic Systems
Off
On
Load
130
120
110
100
90
80
70
60
50
40
30
20
10
0
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
OFF POWER ON
V.HOLD
H.HOLD
BRIGHT
CONT

It's like adding a room to your brain!

The System 8813 from PolyMorphic Systems is a complete, powerful problem solver in a single walnut cabinet. This machine allows you to perform complex financial, engineering, and scientific models in the comfort of your office or den.

The high speed video display presents your results in text, tables and graphics. The detachable typewriter-like keyboard permits relaxed program entry and operation. Convenient mini-discs store programs and data for compact filing, secure storage, and fast access. Our disc BASIC programming language is simple enough for the computer newcomer, yet powerful enough to amaze the most advanced users.

The whole family can immediately use and enjoy pre-programmed applications and educational packages. Let the System 8813 become your trusted business, profession, and personal tool. The PolyMorphic Disc System is a completely assembled and tested unit with brushed aluminum front panel, walnut cover, detachable keyboard, video monitor, 16K RAM. Includes system software and fully extended BASIC on disc.

System with 1 disc drive —	$3250
System with 2 disc drives —	3840
System with 3 disc drives —	4430

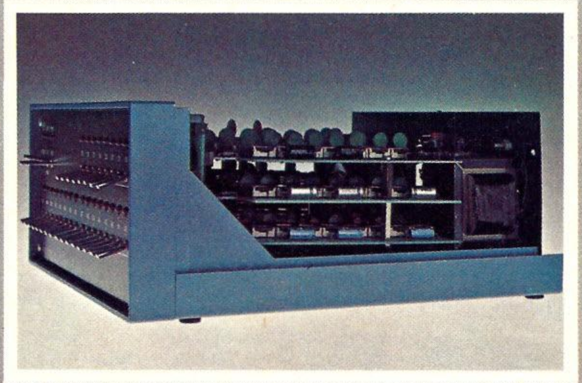
Delivery 60 days ARO. Upgrade packages for POLY 88 owners will also be available. Prices and specifications subject to change without notice.

PolyMorphic Systems
(805) 967-0468
460 Ward Drive, Santa Barbara, CA 93111

Powerful in computing muscle, yet small in physical size, the Altair™680b offers many special features at an affordable price. Based on the 6800 microprocessor, the 680b comes with 1K of static RAM, Serial I/O port, PROM monitor and provisions for 1K of PROM as standard components. It's good thinking, when you're interested in making a modest investment on a highly reliable computer, to consider the Altair 680b.

Our PROM monitor eliminates the necessity for toggling front panel switches to load bootstraps or manipulate memory contents. Only a terminal and programming language are required for complete system operation. With Altair System software—Altair 680 BASIC, assembler and text editor—you may begin problem solving immediately with ease.

By adding the 680b-MB Expander card, many options are currently available:



*16K Static Memory Board—Increase your system memory with 16K bytes of fast access (215 ns), low power (5 watts per board) static RAM. 680 BASIC and assembler/text editor are included free with purchase.

*Process Control Interface—A PC card that uses optically isolated inputs and relay outputs that transmit sensory information to and control signals from the computer. A diverse world of control applications is opened up with the Altair 680b-PCI.

*Universal Input/Output Board—If your I/O needs exceed the serial port already on the main board, augment your I/O channels with the 680b-UI/O. By implementing the optional serial port and two parallel ports, you can simultaneously interface to four terminals.

*New Addition—Kansas City Audio Cassette Interface—Use the 680b-KCACR to interface your Altair 680b with an audio cassette recorder for inexpensive mass storage of programming languages, programs and data.

Available in either full front panel or turnkey models, the Altair 680b presents many computing capabilities at a low cost—without skimping on performance. See it today at your local Altair Computer Center or contact the factory for further details.

mits

Good Thinking.

AC
HLT
RUN
RESET
HLT
DEP
RUN
D7
D6
D5
D4
D3
D2
D1
D0
DATA
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
ADDRESS
MITS ALTAIR 680

2450 Alamo S.E. Albuquerque, New Mexico 87106
dealer inquiries invited.

Volume I, Number 3

Contents

September 1977

FEATURES

18 **Computer Wrestling: The Program of Champions** **by Lee Felsenstein**
Microcomputers are more than a match for complicated score keeping.

26 **Forget Me, Forget Me Not** **by Avery Johnson**
Are we remembering too much? And what's the address for "intention?"

28 **Bits, and the Thin Red Line** **A short story by Robert Abel**
It's a crime. Or soon would have been.

34 **PLATO Makes Learning Mickey Mouse** **by Elisabeth R. Lyman**
A look at the largest up and running educational computer system.

40 **Charged Couples** **by Sandra Faye Carroll**
CCDs: How they work and how they're made.

46 **Xeroxes and Other Hard Copy Off Your CRT** **by Bill Etra**
Shoot it now, see it later.

55 **The Kit and I, Part Two: or Power to the Computer** **by Richard W. Langer**
The continuing saga of one man's fun and frolic assembling a computer.

60 **How Computers Work** **by Joseph Weizenbaum**
A basic tour of operations from one who really knows.

80 **Personally Yours from IBM** **by Eben F. Ostby**
Is the 5100 a home computer?

87 **A Payroll Program for Your Small Business** **by Robert G. Forbes**
Keeping track of Uncle Sam's slice of your pie with a micro.

COLUMNS

10 **Missionary Position by Theodor Nelson**
Upstairs, downstairs, at Dallas.

12 **The Human Factor by Andrew Singer**
Books, books, books.

16 **Legal ROMifications by Peter Feilbogen**
Copyright and your data base.

98 **FutuROMa by Bill Etra**
Your PC can predict the future.

DEPARTMENTS

6 **On the Bus**
8 **Reader Interrupt**
11 **Eve 'n' Parity**
49 **Run On Micros**
50 **Centerfold**
96 **The Noisy Channel**
97 **Babbage and Lovelace**
100 **PROMpuzzle**

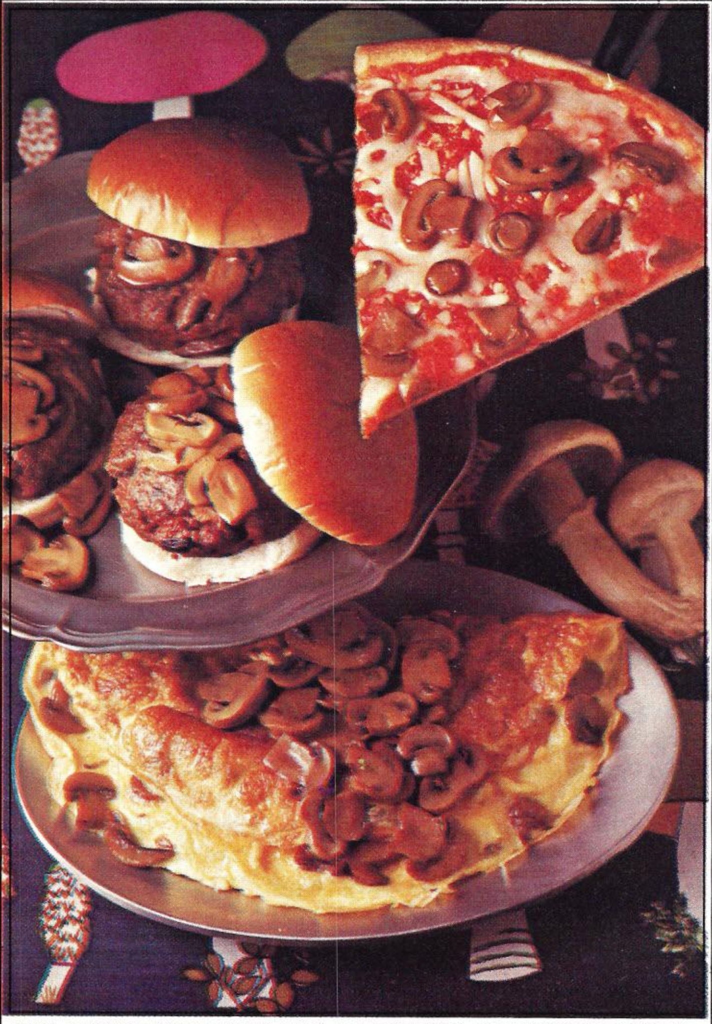
ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). *Copyright © 1977 by ROM Publications Corporation. All rights reserved.* Reproduction in any form or by any means of any portion of this periodical without the written consent of the publisher is strictly prohibited. The following trademarks are pending: Babbage and Lovelace, Eve 'n' Parity, floppyROM, futuROMa, The Human Factor, Legal ROMifications, Missionary Position, The Noisy Channel, On the Bus, PROMpuzzle, Reader Interrupt, ROMdisk, ROMshelf, ROMtutorial, and Run On Micros. Opinions expressed by authors are not necessarily those of *ROM* magazine, its editors, staff, or employees. No warranties or guarantees explicit or implied are intended by publication. Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

COLLIER IS THE BEST ENGRAVER, PRINTER IN THE COUNTRY.

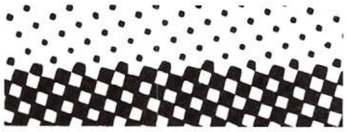
Because. A revolution in engraving has happened. Collier split the dot. And because of the new Laser Beam, Collier is now light years ahead of the rest. The laser does it. Here's how it works. Technically, the laser beam which is used to control film exposure (thus "Program" the engraving) is split into six sectional beams. Each of these is digitally modulated by computer to transfer half-dots of picture information per scanner revolution. Since two picture-information "bits" are available per dot in circumferential direction, it's possible to expose a smaller area in the second "orbit"...or even completely omit it (thus producing an actual half-dot or even an elliptical mini-dot). If that seems to all sound a lot like gobbledygook, don't worry about it. The important thing is, it's here and it does work and it gives your image resolution that you never could get before. We'll be happy to demonstrate it for you. Even happier to produce your next set of four-color letterpress, offset and gravure engravings. Call Collier Graphic Service Company Inc., 240 West 40th Street, New York, New York 10018/(212) 840-0440.

ATLANTA	BOSTON	DETROIT	HARTFORD	NEW YORK	PHILADELPHIA
(404) 892-2383	(617) 965-5660	(313) 259-2111	(203) 367-0706	(212) 840-0440	(215) 988-0110

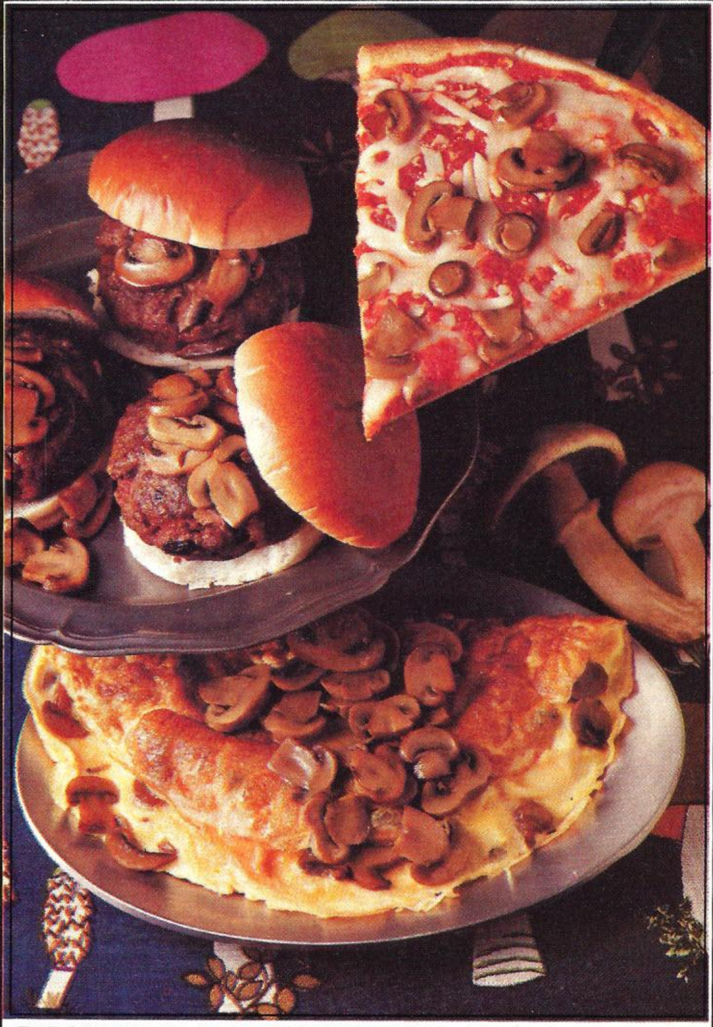
OR CALL TOLL FREE (800) 221-2585/(800) 221-2586



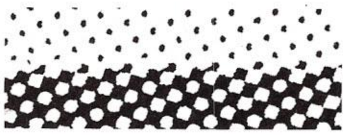
THE ABOVE ENGRAVING WAS MADE WITH THE CONVENTIONAL PLATE MAKING PROCEDURES, AS YOU CAN SEE, IT LACKS COLOR FIDELITY AND SHARPNESS, IF YOU COMPARE IT WITH...



THE CONVENTIONAL ENGRAVING DOT.



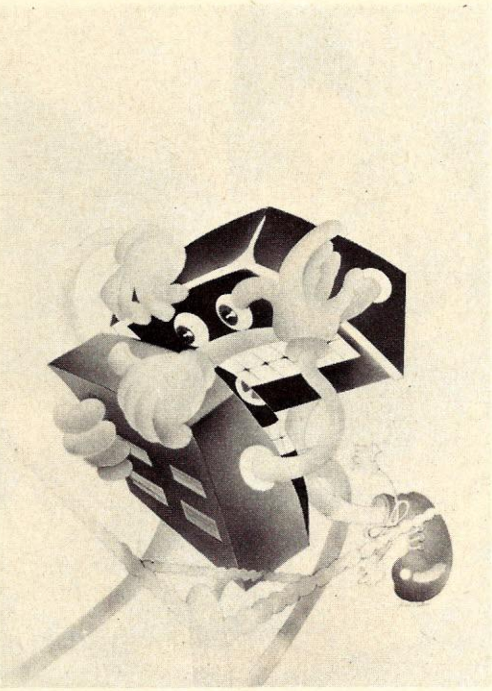
THE COLLIER'S LASER BEAM ENGRAVING, AS YOU CAN SEE FROM THE ABOVE, HAS BRILLIANCE, SHARPNESS AND COLOR FIDELITY THAT ONLY COLLIER'S DOT CAN PROVIDE.



THE COLLIER LASER DOT.

THE COLLIER ADVANTAGE. COLLIER GRAPHIC SERVICES COMPANY, INC., 240 WEST 40TH STREET, NEW YORK, NEW YORK 10018

On the Bus



Rex Ruden started out by drawing cowboys, but he didn't know how to do the feet, so he had to put his favorite guntoters behind rocks or knee-deep in grass in all his early efforts. Since elementary school days, however, feet have ceased to be a problem. His experience has been mostly in display, advertising, and animation. He also worked with well-known black animator Tee Collins. After spending the last five years with Aetna in the company's audio-visual department, Ruden plans to work more and more on magazines and children's illustration.

THE COMPUTER STORE

Featuring

MITS Altair
Data General Micro Nova

Also your headquarters for

Proto Typing Equipment
Wire Wrapping
Magnetic Supplies

"Where personal computing begins."

THE COMPUTER STORE
63 SOUTH MAIN ST.
WINDSOR LOCKS,
CONN. 06096
(203) 627-0188

Robert Abel is a journalist, a small press editor (Lynx House), and has published fiction in a number of small press journals, including *Epoch, Kansas Quarterly, Dark Horse,* and *Colorado State Review.* He is a closet science fiction writer, journalist, and small press editor. A former college teacher, he sometimes thinks he would have been better off as a jockey, but, of course, all that has been ruined by too many beers.

Hooked on crossword puzzles at an early age, **Daniel Alber** now constructs as well as solves them. Part of the Brownstone Renovation generation in New York, when he's not constructing puzzles for the likes of *Field and Stream, The New York Times,* and *ROM,* he's reconstructing olden golden rooms in his Brooklyn based house.

Sandra Faye Carroll is currently a contributing editor and The Underground Gourmet for *New West* in San Francisco. She has written for such diverse publications as *The New York Times Book Review, City, Earth Times,* and *Oui* on subjects ranging from carcinogens and energy issues to ethnic cuisines.

Bill Etra is an Instructor of Home Computing at the New School for Social Research in New York. He is co-inventor of the Rutt/Etra Video Synthesizer—the first portable voltage control analog video synthesizer, as well as the Videolab. His main interest is videographics, and many of his works have appeared as cover illustrations on various periodicals and books including *Computers in Society* and *Broadcast Management and Engineering.* His current research centers on "The Computer as a Compositional Tool for Video."

Peter Feilbogen attended the Rutgers School of Business Administration and Brooklyn Law School. In addition to being an attorney, he is also a Certified Public Accountant. He has been a sole practitioner on Long Island for approximately ten years, and treasurer of Data Information Services, Inc. An avid tennis player, he is currently trying to improve his game with the aid of a computer.

Lee Felsenstein was born in Philadelphia and grew up wanting to be an inventor. Outside of that, he bears no resemblance to W.C. Fields whatsoever. Instrumental in establishing the first experimental public-access information-exchange system in 1972, he is presently engaged in further development in that area of communications. In his spare time he has designed the Pennywhistle 103 modem, the VDM-1 video display module, the Sol terminal/computer, and the VID-80 video display card. Lee was also instrumental in forming the original Homebrew Computer Club and currently serves as its "toastmaster."

While **Robert G. Forbes** was taking a psychology course in college, he was taught how to do statistical analysis with a computer. Psychology fell by the wayside, and he switched his major to computer science. Currently he is a programmer with the May Company. He has written a number of business-oriented programs, both large and small. On the personal computing level, he's waiting for the minis to open up.

Avery Johnson, with a doctorate in electrical engineering from MIT and five years of post-graduate study in neurophysiology, is uniquely qualified to aid in the goal of making "man-relevant engineering a viable way of living and working." His work is with mobility and sensory aids for the blind, communications systems for crippled children, and physiology of communication. He is currently pursuing research in "soft control material" to provide new interfacing technology between man and machine.

Richard W. Langer is a free lance writer whose articles have appeared in such diverse magazines as *New York, Family Circle, House and Garden,* and *Esquire.* Currently he is a columnist with *The New York Times* and at work on his fifth book.

Elisabeth R. Lyman is a Research Assistant Professor with the Computer-based Education Research Laboratory at the University of Illinois. Ms. Lyman has worked with the PLATO project there for fifteen years in all aspects of the program, including courseware development and information storage and retrieval. She trained as a physicist at the University of California in Berkeley.

Theodor Nelson is the author of the classic *Computer Lib/Dream Machines,* a Whole Earth style catalogue of computer machinations. Presently at the Department of Mathematics of Swarthmore College where he is working on the Hypertext Project, Ted specializes in highly interactive systems for graphics and text. His past experience includes a stint at Dr. Lilly's Dolphin Laboratory and work as a consultant for Bell Lab's ABM system.

Eben F. Ostby first began experimenting with computers while at the Pomfret School, which had a PDP-8. He went on to become the first Computer Science major at Vassar College, from which he recently graduated with honors. Ostby has done extensive work with graphics in APL, and recently programmed what he thinks may be the first cartographic project in APL.

Andrew Singer has been hooked on computers since he first built one in 1958. A hard/software consultant, he is fluent in thirty computer languages and knows more than enough about twenty species of machine. His work has included the first medical information retrieval system based on ordinary clinical records, and a large and intricate system for interactive selection of data from public opinion polls. He believes that most software is poorly designed and unspeakably rude, and his Ph.D. research is aimed at improving the architecture and human engineering of interactive systems.

Joseph Weizenbaum is Professor of Computer Science at the Massachusetts Institute of Technology. He is best known to his colleagues as the composer of SLIP, a list-processing computer-language, and for ELIZA, a natural-language processing system. More recently, he has directed his attention to the impact of science and technology—and of the computer in particular—on society. ▼

RAINBOW COMPUTING, INC.

Supplier of

WAVE MATE
THE DIGITAL GROUP
DEC PDP
Computer products

Peripherals and Supplies from

PERSCI
CENTRONIX
DIABLO
MAXELL
COMPUTER DEVICES
LEAR-SIEGLER
MULTI-TECH
TEXAS INSTRUMENTS

Specialists in Design, Implementation
and Support of
Custom Hardware/Software Systems for
Business, Educational, and Personal Use

Experts in most major computer
software including
CDC, IBM, PDP
BASIC, COBOL, FORTRAN, PL1
LISP, SIMULA, SNOBOL, SPSS, BMD's
COMPASS, MACRO,
6800, & Z80 assembly languages

10723 White Oak Ave., Granada Hills,
Ca. 91344
(213) 360-2171

Reconditioned Teletypes

Guaranteed 90-days on return basis

ASCR 33 $950 KSR 33 $595

Free delivery within 25 mile

radius of New York City.

Shipping extra beyond radius.

IMSAI edge connector and guides:

ten for $39, $4.50 each.

COMPUTER MART OF

NEW YORK

118 MADISON AVE

NEW YORK, NEW YORK 10016

(212) 686-7923

Reader Interrupt

Your letters with their reactions to our first issues are beginning to come in. (Those we printed in the second issue were in response to dummy copies sent out in a sampling—the lead time between letters and their appearance in print is normally two to three months in a periodical that's published twelve times a year.) On the whole, the letters are favorable, which is nice. But favorable opinions alone aren't really much to publish in a letters column. What we need is some controversy and some questions—which I'm sure we'll get soon enough.

Meanwhile....

You'll notice our September issue comes out in September. Other magazines dated September come out in August. In some extreme cases they are put on the stands as early as July. We had every intention of playing the same game as everyone else. Unfortunately, we ran into some two hundred National Guardsmen, a missing truckload of *ROM*s, assorted bloodied noses, bashed heads, and arrests. You guessed it. Time for that traditional American pastime—the negotiation of a labor contract at the printer's.

Our printer, Banta Company, of Menasha, Wisconsin, was shut down by a strike just as we were getting ready to put out the first issue of *ROM*. Although we were a very small cog in a large machine, and had nothing directly to do with the strike (except, perhaps, that we choose to print with a union shop), *ROM* was temporarily crushed.

The printing plates were inside, the unions were outside, and *ROM* was nowhere. Eventually, Banta did manage to shunt us off to another printer for the first issue, but they had no binding facilities. Nor any sheet-fed presses to print the cover and centerfold. So part of *ROM* went off to another printer and part of *ROM* went off to still another printer. Eventually, they met, but at a printer which had no wrapping facilities for mailing. And who didn't have time to package the bulk in cartons for shipment to magazine wholesalers and computer stores.

So part of *ROM* went off to yet another printer for wrapping and it turns out that....

To make a long story short, there were *ROM*s running back and forth all over the state of Wisconsin for the better part of the month. With us running right behind them since the trucks kept failing to ship on their appointed days. Now maybe the sensible thing would have been to switch printers. On the other hand, that seemed like deserting a friend, so we stuck with the struck.

With all this, the July issue, which was supposed to come out in June, came out in July. The August issue, which was supposed to come out in July, came out in August. This September issue, which was supposed to come out in August will, hopefully, be in your hands at least by the middle of August. And....

What does this all mean? Who really knows? It's just you can't come out the same month that's on the cover if no one else does.

Dear ROM:

I still have not received my July issue. Please follow up on this matter and let me know as soon as possible if *ROM* was actually sent out yet or if it was lost in the mail.

John S. Wood
Chicago, IL.

Dear ROM:

I haven't received my July issue. Please contact me and let me know if you have received my request for a subscription.

Joseph E. Wilson
Philadelphia, PA.

See what we mean. Nobody expects July in July any more. Maybe it's like all the stores putting up Christmas decorations the week after Halloween.

Whatever. If you really need your October issue of *ROM* in September, well, we can always skip a month to catch up some time—after the strike is over. ▼

Editor and Publisher
Erik Sandberg-Diment

West Coast Editor
Lee Felsenstein

Associate Editor
Janet C. Robertson

Contributing Editors
Sandra Faye Carroll
Bill Etra
Louise Etra
Ed Hershberger
Avery Johnson
Richard W. Langer
Theodor Nelson
Robert Osband
Frederik Pohl
Andrew Singer
Alvin Toffler

Crossword Puzzle Editor
Daniel Alber

Art Director
Susan Reid

Contributing Artists
Robert Grossman
Cindy Hain
Luis Jimenez
Korkie
David Macaulay
Rex Ruden
Linda Smythe

Art Assistant
Sue Bass

Staff Photographer
Thomas Hall

Composition
Lynn Archer

Editorial Assistant
Donna Parson

Special Assistants
Jennifer L. Burr
Joanne Zeiger

Counsel
Peter Feilbogen

A Note to Contibutors:
ROM is always looking for good computer applications articles from people with up-and-running systems. We also will be glad to consider for possible publication manuscripts, drawings, and photographs on other computer-related subjects. Manuscripts should be typewritten double-spaced, and a stamped self-addressed envelope of the appropriate size should accompany each unsolicited submission. Although we cannot assume responsibility for loss or damage, all material will be treated with care while in our hands. Contributions should be sent to ROM Publications Corporation, Rte. 97, Hampton, CT 06247.

You're curious enough to read ROM now, you'll be curious enough to read ROM every month. So subscribe!

SAVE $ 7.00 over the single-copy price with a one-year subscription

SAVE $20.00 over the single-copy price with a two-year subscription

SAVE $33.00 over the single-copy price with a three-year subscription

SAVE EVEN MORE with the Rich Uncle Special:

For the first six months, and the first six months only, ROM **is offering genuine inflation-proof lifetime subscriptions at $250.00 each. No matter what happens to the dollar compared to the yen, the mark, the franc, or even the yak, with the Rich Uncle Special you're assured of a lifetime supply of** ROMs. **At today's rate of inflation, a year's subscription may well be selling for that much in only a decade.**

GUARANTEE :
If not satisfied with **ROM** at any time, let us know. We'll cancel your subscription and mail you a full refund on all copies still due you.

ROM
COMPUTER APPLICATIONS FOR LIVING

ROM Publications Corp.
Route 97
Hampton, CT 06247

Name ______________________

Address ______________________

City ______________________

State __________ Zip __________

☐One year $15 ☐Two years $28 ☐Three years $39
☐Check or money order enclosed. ☐Lifetime $250
☐Master Charge ☐BankAmericard
Exp. date Card# ______________________

Please allow 4-6 weeks for delivery.

Missionary Position

UPSTAIRS DOWNSTAIRS

by Theodor Nelson

Once a year, everybody in the computer field goes to some weird place, and they exchange brochures and say peculiar things in crowded rooms, and that is called the National Computer Conference.

This year it was in Dallas, and 35,000 people came, and lo there was much literature exchanged and beer swug and hot dogs et. And finally the national organizations that hold this shindig got around to recognizing that yes, there is something called Personal Computing. Portia Isaacson, who chaired the NCC this year (and also has a computer store in Dallas), made them include personal computing in the National Computer Conference.

The computer profession at last had to acknowledge the computer non-profession. Square computing met amateur computing head-on, and nothing happened.

This was my fourteenth National Computer Conference (counting the Joint Computer Conferences that came before), but for the last couple of years the action has been so heavy in personal computing that I'd almost lost the feel of what it's like in square computing, back where a mainframe is still a million bucks, or ten thousand. Here the two were side by side, making for a funny experience: you could pop back and forth between the two worlds, rather like an actor or a spy, experiencing the sudden feel of each repeatedly.

The main commercial exhibits were upstairs.

The personal exhibits were downstairs.

And it was just like in the British TV soap opera.

Upstairs they are haughty and pompous, and have no idea of how life is lived below. Downstairs they know what really goes on upstairs, and what closets the skeletons are in.

Upstairs they think the old ways will go on forever.

Downstairs they weep at the human consequences of the foolishness above, and pick up the pieces.

The big news upstairs, I think, was laser printers, and the "Pray for IBM" buttons that *Computerworld* was giving out. (Will anybody sell me one?) The rest was terminals and ever-improving, unexciting whatnot.

But the news downstairs will affect millions of people, which laser printers won't.

The big news downstairs was still the same as the big news in California in April—the Commodore Pet computer for six hundred bucks, including screen and keyboard and ROM BASIC. Except now they had production prototypes on the floor as well as the wooden model from California.

It was the biggest news on the grapevine in March; it was the biggest news at the West Coast Computer Faire in April—especially because it worked, though its rounded "futuristic" housing was plainly hand-shaped of wood—and it's still the biggest news, because now the cabinets seem to be production plastic and there's an order blank. Your $595 check must accompany coupon. It looks real.

I think they'll sell millions.

Commodore's strategy is very clever. They've acquired MOS Technology, makers of the 6500 chips and the KIM. The 6500 is inside the Pet. But the Pet Computer they sell for $595 is just the beginning, with its 4K of open memory, screen and keyboard and IEEE bus and ROM BASIC. Now for the bad news. The next 4K is another $200.

More software will come in ROM blocks, presumably blisterpacked, to be sold like Hot Wheels and Barbie Dolls. Commodore folks say they'll be delighted to publish software—as ROM blocks.

If the $50-a-K price for memory is any indication, though, the software is not gonna come cheap. In other words, they think they've got it boxed in.

So does Fairchild, with the Channel F. Channel F, to the public, is a red hot ziggety Video Game that hooks to your TV and costs $150. With a few games built in, it also takes a four-track tape cartridge that loads in *other* video games. At twenty bucks a cartridge (list).

That's the Channel F as the public sees it and as Fairchild presents it. As you and I see it, though, it's actually a dinky computer: inside is actually the Fairchild F-8 computer chip, several K of memory, a Dazzler-type bit-map display generator, and of course your video modulator that goes in through the antenna.

Wellsir! As I said in my after-dinner harangue at the Faire, there's a swell opportunity there for someone who wants to break open the Fairchild and help it be all that it is.

Just that has happened. Someone's broken into Fairchild's little Video Game and made out of it the computer it really ached to be.

Fairchild had talked as if you couldn't, but certainly didn't take any precautions: you scarcely have to do more than plug something into a socket that's already there to get a functioning F-8.

One booth at the NCC, Downstairs, was a father-and-son team who took the cover off the Channel F and found nice places to put a UART, making the device right off the bat into either a computer or a computer-driven graphics display. On top of which, a computer store says it will shortly offer a conversion kit.

This is really fun. Here is Fairchild, high and mighty, thinking it's got a lock on the box and only *its* chosen uses can be made of the product. Off comes the cover and ha ha ha, look what we can make it do, against Fairchild's wishes. But of course the last laugh will be Fairchild's, since to convert the box you've got to buy it. Even if they didn't see in the first place where home computers are going, there will be many hands, and conversion kits, to show them the way. (Certainly somebody else will be first to offer the box that *writes* tape cartridges for the Channel F.)

Which brings us back to Commodore. The Pet comes with its cover on, and the warranty will presumably say hands off. But how long will it be before the Pet has been opened, explored, adapted and kitted? How long till you can add S-100 bus, music and the rest?

Les Solomon, editor of *Popular Electronics,* gives it a couple of months. He expects a black market in Pet schematics before Christmas.

If the Commodore is good—and their stuff tends to be (I bought a printing calculator from them eight years ago, and it's still the best)—then this unit will be like the Volkswagen: a low-cost workhorse for many, chopped and channeled California style for many others.

Every year there has been *the* personal machine. In Year One of personal computing, 1975, it was of course the Altair. In Year Two it was the Imsai. This year it's the Sol. What's next, the Pet?

Hard to say. I don't think there will be "the" single *au courant* machine from now on, because the market is going to go so many ways.

The LSI-11, if indeed Heath is putting one out, will certainly be the top machine. But there will also be the S-100 world continuing, and below that the funny little dingbat programmables, and on a new branch of the tree the Pet-style machines.

(By next year the emphasis will have gone over to software anyhow.)

Some people from straight computing are tuning in to all this. Quite a few did come down to the personal computing exhibition. But a lot of people didn't.

Quite a few people, too, came to the personal computing sessions, but rather few—say, a few thousand, compared to the overall 35K that came to the NCC. Those that came to the personal sessions seemed an interesting menagerie: there were those already in the field; the curious, who might get into it; and others who had obviously been *sent*. Worried-looking, confused, usually middle-aged, these guys took the most notes; it seemed to me that their function was to report back to somebody what personal computing was about. But from the faces, and the style of note-taking, it was clear that the big picture wasn't getting across: those who were sent to report back won't be able to put it into words that their bosses can understand.

I went to a very nice buffet breakfast for the speakers. Two other guys were at my table, and they couldn't *imagine* what personal computers could possibly be for. They worked in large-scale simulation: the computer for them is something at the office that they work on, like a drill press.

In the text system session, I heard people tell how they use computers to get a lot of stuff printed out. I asked what ideas they had for personal writing systems. One panelist said he couldn't imagine anyone putting up funds for such a thing. Obviously he hadn't been downstairs.

In a session on small business systems, panelists told us that a small business system will cost about $40,000 for the hardware and $40,000 for the software. From the floor I asked for a show of hands as to who was working on software for the *cheap* machines: at least twenty-five hands went up.

From all these things I came away with this impression: the straight computing world still doesn't even know about its bastard child, its demon seed. I guess they think "personal computing" means merely the reclaiming of scrap machines, or doing odd programming tricks with pocket calculators. Something akin, in scope and profundity, to soap carving or toothpick model-building.

In a way that's true. People do do things like that, weird dead-end tricks, and in that frame of mind. And there are those who just like to solder.

But there's a much grander vision, and I think most of us in personal computing share it. It's the idea that your computer, yours, your truly own, will make your life better and simpler, easier and deeper and richer, both by taking care of your old problems and by providing you with new and wonderful things to do, alone or with those you love.

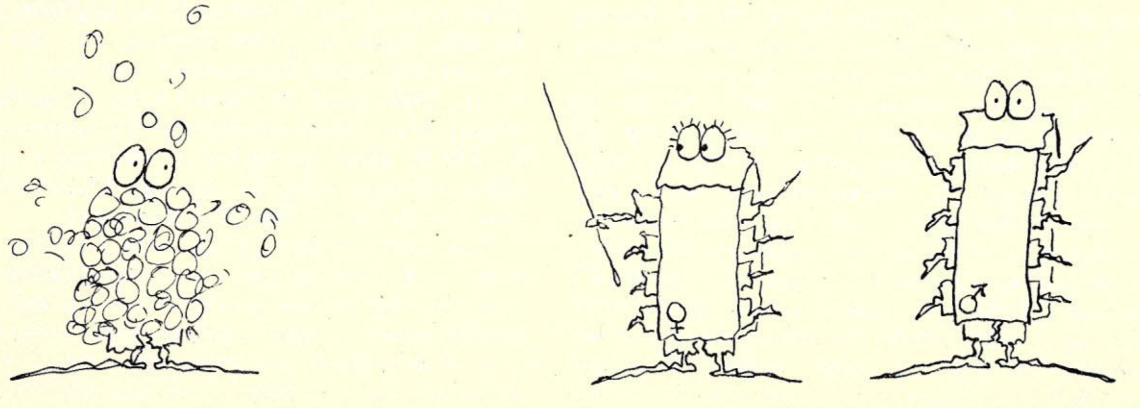
The computer will simply be a personal super-secretary and nerve center and visualization tool, for finances, writing, planning, idea-keeping. Taking the drudgery out of all these things and making everything simple. And an interactive movie-box that takes you to important worlds you must understand.

Words don't seem to help explain this to most people, though. There seems to be a certain necessary vision. Either it is all quite obvious to you or you will never be able to figure it out. Maybe in a while, when a good panoramic personal software system exists, it will be easy for anybody to get the idea, but it isn't yet. Meanwhile a lot of us just continue advancing slowly on the dream, on hands and knees.

What can you use a computer for in the home? Tell the squares you can pound nails with it. And hang in.

It may be that the distance between old-style computing and personal computing is narrowing, that they are drawing closer together. But it looks to me like the Titanic and the iceberg. ▼

EVE 'N' PARITY



"I'm going to puncture that bubble memory's ego bit by bit."

Cartoon by Korkie

The Human Factor

BOOKED UP FOR FALL

by Andrew Singer

From time to time, people ask me to suggest books that they can read to help them understand computers. Thousands of books have been written about computers, but few are fit for ordinary human consumption. As a reader, I strongly favor authors who use English. This is a rarity in the computer field, where a book is just as likely to be written or encoded in some variety of computer language or mathematics. I go out of my way to avoid impenetrable books, and this saves me a good deal of time, but a diversity of interests manages to keep me pretty busy in the library, just the same. An intriguing book is also apt to inspire in me an impertinent curiosity about its writer, which I try to satisfy as the occasion permits.

What follows, therefore, is a sort of book salad, sprinkled liberally with the oil and vinegar of my observations about the responsible parties. It is neither exhaustive nor conclusive. A warning: I may decide next week that one or another of the books described is actually pretty awful. But this week they are all in favor, and I recommend them without exception (except as noted).

Ted Nelson, compulsive coiner of words and a contributing editor to this magazine, is probably best described in his own words as "chronically afflicted with idearrhea." This continuous inspiration is evident on every page of *Computer Lib/Dream Machines* (available almost anywhere, Ted Nelson, publisher—soon to be renamed The Press of Circumstances—$7.) Although it has been described as the Whole Earth Catalog of computers, I believe this is inaccurate. Perhaps Whole Alpha Centauri Catalog would be better, certainly for the *Dream Machines* section which is definitely "Star Wars. Contemporary." I should point out that this book is really two books. *Computer Lib* is a non-threatening introduction to computers, large and small, their uses, users, and vernacular. The flip side of the book, *Dream Machines,* is a remarkable tapestry of reality, fantasy, and showmanship, picturing the alternative worlds of future and not-so-future computing. Incredibly, Ted was proposing many of these same concepts when I first met him in 1966. People take him a little more seriously now, but it will probably be another decade or two before most of his ideas are finally realized. This book possesses a humor and humanity exceedingly rare for the computer field.

Although *Computer Lib* is written in a non-technical style, the book is nevertheless technically accurate. Ted's analysis of IBM alone is probably worth the price of the book. Nitpicking, I might point out that the author's dedication to the computer language, TRAC, is probably a bit misplaced, but he and I have disagreed about this for years.

From a Human Factor standpoint, the book has one significant flaw; the type size has been borrowed from the three-inch, shirt-pocket edition of the King James Bible, so bring your magnifying glass. Periodically, Ted assures me that this defect will be corrected in an upcoming edition, but I doubt it. I think he enjoys the idea of all those readers poring over his text like Sherlock Holmes, searching for a clue to the mystery of the computer.

Electronic Design is a magazine which is mailed free to those deemed by its circulation department to have sufficient purchasing clout in the electronics industry to make the giveaway worthwhile. If you don't qualify, you can subscribe for thirty dollars a year (maybe higher by now). But if you are only interested in a selected area like microprocessors, you're better off waiting and buying a collection of topical articles periodically published by Hayden Book Co., *ED*'s sister company.

Microprocessors, New Directions for Designers ($10.95) is just such a collection as is the more recent *Microprocessor Basics* ($9.95). Articles range from the highly technical to the technically banal, and most deal with commercial applications of microcomputers, especially from a hardware point of view. If you like to get your hands dirty diddling with flip-flops and gates, you'll find these books provocative. I particularly enjoyed the article on motor control by a microprocessor in *Microprocessors, NDFD* and the detailed comparison of various microprocessor architectures presented in *Microprocessor Basics.* If terms like "flip-flops," "gates," and "microprocessor architecture" have already left you in the dust, these books are probably not going to be too helpful.

Because most of the articles are written or rewritten for the general engineering audience, usually by professional writers, the writing quality is above average. But don't expect much style. The most valuable aspect of these books is that they accurately reflect the state of the art in microprocessor technology. If you're not interested in that, you won't be captivated. Not recommended reading for hardware unenthusiasts.

I have no intelligence on John F. Young other than what is on the jacket of his book, *Robotics.* He's an electronics engineer, a manager, and a lecturer in electronics at the University of Aston in Birmingham, England, active in cybernetic engineering and robotics, etc. But if you are interested in robots, *Robotics* is a book you'll want. It's not cheap. Distributed by Halsted Press (a division of John Wiley), it's available in academic bookstores in hardcover for $18.95. Don't let the high price deter you. This book offers the most complete coverage I have seen so far of what's been done in the field of robotics.

The book is broken up into topic areas like vision, mobility, and reliability, and it contains detailed engineering specifications for human capabilities as well as those of existing robotic systems. Special chapters deal with particular robot-related problems like power sources and actuators. Every chapter has an extensive bibliography. I was intrigued by Young's description of some of his own work, including a simple robot tortoise called Astor which is described at the end of the chapter on "Robot Stability."

Robotics is a highly technical book, but the writing and explanations are good enough, so that even the general reader should be able to get an overview of the field. Highly recommended for anyone interested in real androids. Especially recommended for budding your Rossums. Don't expect to read it in one sitting.

Donald Knuth has been active in the computer field since he started writing *Tic-Tac-Toe* programs for the Case Tech IBM 650 in the late 1950s. I've only met Knuth a few times, but he was already a legend around Case by 1960 when I was an undergraduate there. His seven-volume *The Art of Computer Programming* (Addison-Wesley, available in academic bookstores, $20.95 per volume) was intended to be *the* definitive work on the subject. Unfortunately, the computer field changes so fast that it can only be considered as *a* definitive work. All the volumes written so far (three) are excellent, but I especially recommend Volume II, *Semi-Numerical Algorithms.*

These books, while clearly written, may be too mathematical for the general reader. It takes some getting into, but Knuth is really worth the effort. Whether you want to find the day that Easter will fall on in 2077, or a good technique for generating random numbers, it's all in there, well-indexed and very thoroughly explored. A friend of mine and former classmate of Knuth's has suggested that a more appropriate title for the series might have been, *Things That Interest Me by Donald Knuth.*

These days, Knuth has been seen riding elevators around Stanford University, where he teaches and continues to add volumes to his series. These are reference works for the serious programmer and problem solver. Not strongly recommended for the naive, all others pay cash.

Numerical analysis embraces the art and science of getting a computer to perform mathematical calculations in a satisfactory manner. Because of time and space limitations, computers don't do arithmetic perfectly, and so there are numerical analysts to smooth out the rough edges. If someone had asked me a year ago what to read to get up on numerical methods, I'd have looked depressed, sighed, and unhappily suggested some obscure mathematician's delight. But now we have Jon Smith's *Scientific Analysis on the Pocket Calculator* (Wiley-Interscience, $12.95 from academic bookstores), a book with wider application than its title suggests.

Although the first chapter seems to deal strictly with pocket calculators, many of the observations in it are equally relevant to algebraic computer languages such as BASIC, APL, and FORTRAN. The rest of the book, though slanted toward the calculator user, is more than satisfactory as a general introduction to numerical methods. In fact, it's so much more than satisfactory, that I can't think of anything better.

While considerations of keystroke efficiency may not be too interesting to the small-computer user, almost everything else in the book will be. The basic methods of the numerical analyst: interpolation, extrapolation, successive approximation, series, Runge Kutta, and difference equations, are presented clearly and with sufficient generality to show the scope of their application. At the same time, discussions of relative and absolute error, convergence, error propagation, and recursive relations provide insights into the nature of computational methods in general. On top of it all, the book is well-written and comprehensive.

On the drawback side, the author often assumes so much familiarity with mathematics and mathematical notation that the general reader will be lost. For the more advanced methods, considerable mathematical sophistication is required of the reader, but then, what naive reader will be concerned with such methods?

All in all, I recommend the book highly even for the math tyro, providing he can find a friendly mathematician's shoulder to cry on. Turn the front dust jacket upside down for an unexpected surprise.

Post-Watergate morality demands that disclaimers precede my last recommendations, which are probably tainted.

E & L Instruments, Inc. is a company I consult for from time to time. Recently, I taught some seminars for E & L which brought me into close contact with their *BugBook* series (I, II, III, etc., available in most computer stores; prices vary depending on book, $8-$15) written by Messrs. Titus, Larsen, and Rony of Virginia Polytechnic Institute. Don't let the academic background of the principals put you off. Based on my seminar experiences, I would say that the *BugBooks* are the only books I've seen so far that really take you by the hand through the basics of digital logic and microcomputer hardware. While the style is a bit clumsy, and the attempts at humor are awful, the books are written plainly, so a reasonably intelligent person can go through them without the aid of an instructor. Especially high marks go to the authors for the lengths they take to avoid assumptions and jargon until they have thoroughly explained them. Anyone who has tried to write a basic introductory text knows just how difficult this is.

The books are particularly well-suited to the hobbyist or experimenter because they are heavily practice-oriented.

Nothing is ever a rose garden, and the *BugBooks* have their share of thorns. I think they are poorly organized, in the sense that they do not give the learner an effective overview. Generalizing from some of the examples is difficult, and the material is often uneven. For example, *BugBooks* III, V, and VI do an excellent job on microcomputer hardware and interfacing but leave much to be desired on the software front. This is understandable and probably reflects the authors' hardware background and orientation, but it isn't much help for the reader. Finally, the books are written to be used with E & L's line of hardware which is an advantage or disadvantage depending on how you look at it. Certainly, it's an advantage for E & L.

In summary, the *BugBooks* are highly recommended for anyone who knows nothing about computer hardware but has a burning desire to get his hands into it. Late intelligence brings me the following: Beginning with the fall 1977 printings, all the *BugBooks* will incorporate an index to increase their usefulness as reference material. Copies of the index alone will be available for a nominal fee (postage and handling) for owners of earlier editions who want this feature.

These days, I don't get to see much of Bob Glorioso. As a supervisor of applied research and development at Digital Equipment Corporation, he gets to work "any eighty hours a week he pleases" on, as he puts it, "all the things that interest me most." He can't tell me much about what he's doing, but I suspect that if DEC ever goes directly into the consumer market, Bob will certainly have had a hand in it. An extraordinary computer engineer and teacher, so many of his former students are scattered throughout DEC that he can scarcely turn around without running into one of them.

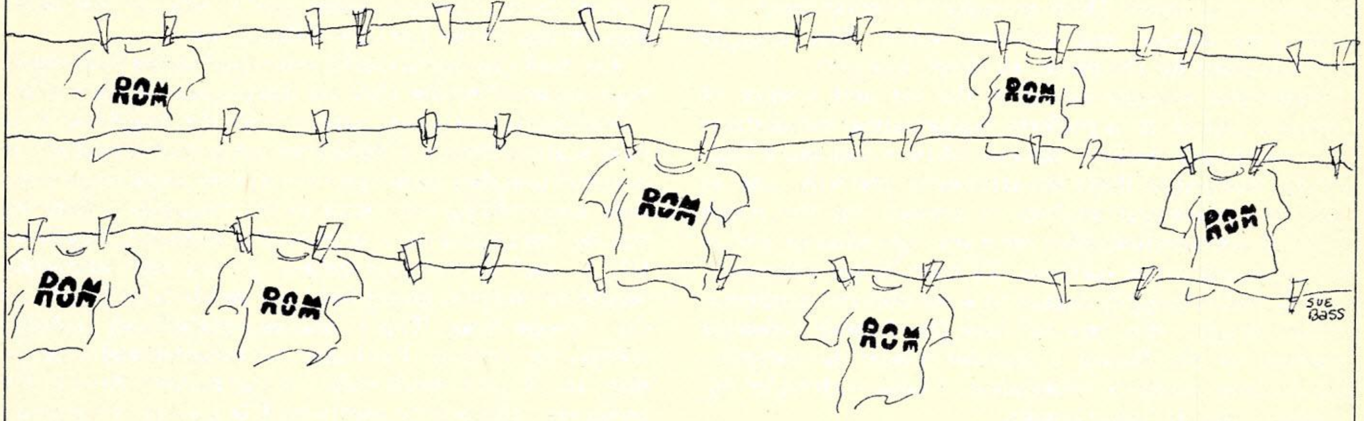
Glorioso's book, *Engineering Cybernetics* (Prentice-Hall, $16.95), is probably somewhat ahead of its time in this country, although he tells me it's selling well in Japan. Although it is a technical book written for seniors or first year graduate students in electrical engineering, I have included it because it is one of the few clearly written books that presents a survey of cybernetic techniques for practical application. Anyone interested in building robots or chess-playing computers will find this book a helpful introduction to cybernetics.

On the defect side, some of the mathematics and terminology may be difficult, and the textbook style won't exactly perk you up. Recommended primarily for the hobbyists and experimenters who are building a HAL 9000 in their garage. Or for professional EEs who are curious about what new technology we can expect from the Honda Household Helpers.

This last recommendation is a pleasure to make, but perhaps I should include a disclaimer. Henry Ledgard has been a teacher of mine, and my research colleague for the last few years. More than most people I have worked with, he shares my concerns for craftsmanship and humanity in the computer field. I believe his books reflect that, and I strongly recommend them to anyone who wants a feeling for the aesthetics and discipline of computer programming. The *Programming Proverbs* series (*Programming Proverbs, FORTRAN Programming Proverbs,* etc., Hayden, available in most academic bookstores and computer stores, $6.50 each) is aimed at teaching programmers good working habits and a respect for their craft. *COBOL With Style* (same publisher, $5.45), and the soon to be available *FORTRAN With Style,* and *BASIC With Style,* refine these ideas further, while introducing superior methods of program design and construction. The "With Style" series owes a great deal to Henry's co-author Louis Chmura whose droll tales of "Irene and the Captain" do much to enliven the books. I liked the author's suggestion in *COBOL With Style,* to "take the afternoon off."

These books are often witty and elegant as one would expect of work which aims to make the case for quality. Highly recommended for the programming enthusiast. A knowledge of programming is necessary, but the English is very plain indeed.

Well, I've come to the end of this column, but since I'm nowhere near the end of my book list, the rest will just have to wait their turn. In the meantime, I guess I'll take the afternoon off. ▼



Get Your ROM T-Shirt Now!

Order from:
ROM Publications Corp.
Route 97
Hampton, CT 06247

Please ship to:
Name__________
Address__________
City__________
State_____ Zip__________

☐S Qty._____
☐M Qty._____
☐L Qty._____
☐XL Qty._____
☐Child S Qty._____
☐Child L Qty._____
☐Check or money order enclosed.
$5 ppd.
☐Master Charge ☐BankAmericard
Exp. date Card#
2 for $9 ppd.
Please allow 4 to 6 weeks for delivery.

New kid on the block!

But watch out he means business

PERSONAL COMPUTING EXPO COMES TO NEW YORK FOR BIG BUSINESS

It's a brand new show in the world's biggest economic center specifically for manufacturers and buyers who are into personal computing. For the first time, this booming field will have a New York Coliseum showcase in the major population center in the east. It is planned as the largest public show of its type in the world that will attract enthusiastic buyers from a multi-state area.

WHY NEW YORK?

New York is the economic nerve center of the world. It also is the world's communications focal point, the one place that will put personal computing in a significant spotlight. New York is surrounded in depth by people who work in the computer field, by computer learning centers, universities, personal computing clubs, and thousands of others whose lives are affected by computers.

From this vast potential, Personal Computing Expo will draw the hard-core hobbyist, the interested student, and, because of a highly-publicized program of introductory seminars, those who are attracted and fascinated by computing but have not had exposure to the ways and means of becoming personally involved.

SHOW MANAGEMENT

Personal Computing Expo is being produced by H.A. Bruno & Associates, Inc., a firm in the exposition and promotion fields since 1923. Highly skilled in the production and promotion of consumer and trade shows, the company currently promotes the American Energy Expo, the National Boat Show, Auto Expo/New York. Promotion assistance also is currently rendered to the National Computer Conference and the Triennial IFIPS Congress in Toronto.

The show producer has promoted successful shows in the New York Coliseum every year since the building opened in 1957. Staff personnel are thoroughly familiar with the building, its services, management and labor.

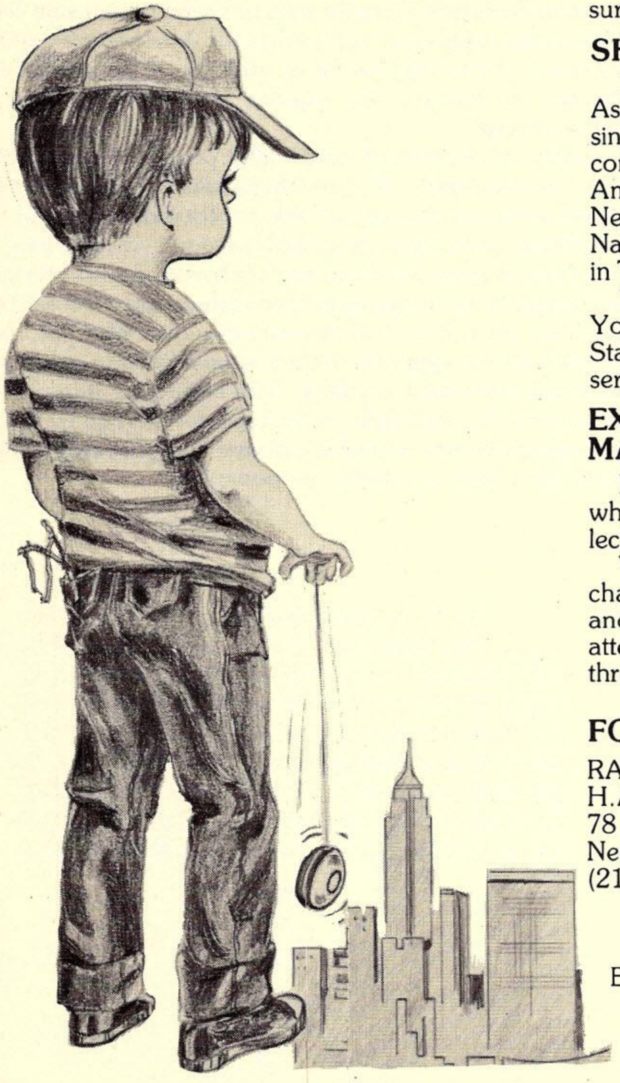
EXCITING SEMINARS FROM "BYTE" MAGAZINE

Personal Computing Expo is endorsed by "Byte" magazine, whose staff is developing an exciting series of seminars and lectures for the exposition.

Visitors to the show will be able to attend these meetings free of charge. They will hear from lecturers such as Louis E. Frenzell and Carl L. Holder. More importantly, visitors will be able to attend meetings aimed at their proficiency levels, from beginner through intermediate and advanced personal computing.

FOR DETAILED INFORMATION CONTACT:

RALPH IANUZZI, Show Manager
H.A. BRUNO & ASSOCIATES, INC.
78 E. 56th Street
New York, N.Y. 10022
(212) 753-4920



Endorsed by BYTE Magazine

OCTOBER 28, 29, 30, 1977

PCE PERSONAL COMPUTING EXPO • NEW YORK COLISEUM

Legal ROMifications

COPYRIGHT YOUR DATA FILE?

by
Peter
Feilbogen,
attorney at law

In broad, general terms, a data base is subject to protection under the copyright law. Because data bases are stored in many ways and are subject to retrieval through many systems, often non-computerized, the copyright laws covering their protection must perforce be broad in scope.

Consider the telephone directory, one of the most common data bases although perhaps not thought of as such by the average user. Printed and bound into a book as hard copy and distributed to subscribers, the telephone directory has long been copyrighted. Many other directories, listings, catalogues, and commercial compilations have a long history of past copyright. Although the technology behind a computerized data base is very different, individuals and firms who create data bases which are used in hard copy face the same problem as the computerized data bank—not to mention the same one that authors and composers have faced for centuries—plagiarism.

However, because of that difference in technology, the solution, indeed the very question as to whether to copyright or not, is not the same. The legitimate user of a data base is not normally interested in the entire file. This user interrogates the base for specific information. Although the total file is available at one time, its bulk is rarely accessed at one inquiry. The danger that such a data base may be plagiarized is substantially reduced by its very use. Is it then worth copyrighting?

The owner who wishes to copyright this property must deposit the *entire* work with the United States Copyright Office. This deposit requirement alone should give data base owners second thoughts on copyrighting their property, for one of the cornerstones upon which a computer data base is founded is the ability to update the file quickly and make the information available to the user. If copyright protection is sought, how often must the updated file be deposited with the Copyright Office?

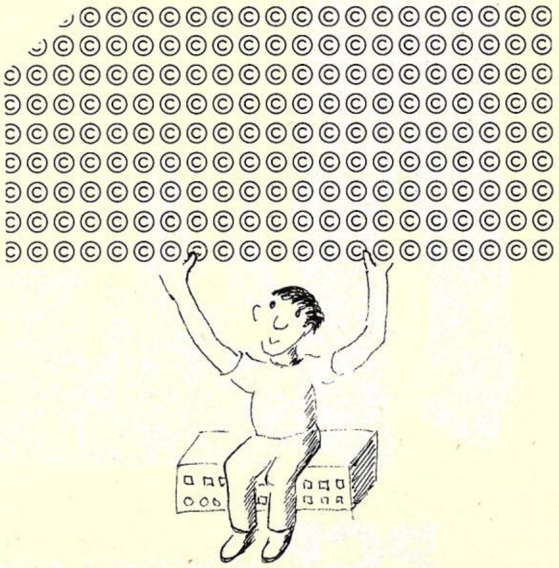
On the other side, updating itself is potent protection. A data base which is constantly changing is constantly outdating itself, becoming progressively more useless to the plagiarizer.

Considering this, for most applications copyright is worthwhile only for relatively static data bases. Fluid data bases will probably not benefit greatly from such protection.

As a personal and home computer enthusiast, the size of your data base would probably be quite small compared to that of, say, a credit card company. Your financial resources are likely to be of even lesser proportions. Do you really wish to incur the expense of obtaining a copyright, particularly considering that you must deposit your work with the Copyright Office for all to see? If you can distribute information in small units, without giving access to the entire file (or documentation), life may be a lot simpler because you build your control over the information into your method of distribution rather than by seeking copyright protection.

Where copyright protection is quite important is if your entire file or great portions of it are distributed. There, however, the question of copyright enforcement, its feasability, and its financial implication becomes crucial. You must also consider whether the information you have compiled and the use it may be put to are broad enough to require protection. Although a merchant's customer list may be useful to another merchant trying to sell to the same socio-economic group, or a magazine subscription list may be useful in someone else's direct sales campaign, and a scientific data base assembled by you could be very useful to students, is the market big enough to warrant the expense and complications of copyrighting?

Lastly, some points of law on other people's data bases for your consideration: If a merchant asks you to convert his typed customer list to tape or disk, the list is still owned by the merchant. Even if you process his accounts weekly from his list, you do not own the data base. If a friend of yours gets hold of a magazine subscription list and wishes to give it or sell it to you, he's not a friend. If the scientific data base you have comes from the appendix of the text book you are using, you have not assembled anything. The important thing to remember is that copyright protection is for original work and belongs to whoever created it, unless transfers or assignments have been filed to modify the ownership. ▼



Illustrated by Sue Bass

IT TOOK THE MEDIA TO KEEP THE GOVERNMENT HONEST. BUT IT TAKES "MORE" TO KEEP THE MEDIA HONEST.

MORE watches the media while the media watches you

What if the news reporters, TV commentators, gossip columnists and media gurus who helped write the Watergate story did as thorough a job investigating their own business?

What if Woodward and Bernstein found a Deep Throat somewhere in the bowels of the Times?

Or if Barbara Walters put Barbara Walters under the microscope?

That's the kind of thing that happens each month in the pages of MORE, the Media Magazine.

MORE covers the media like the media itself covers a big story. By looking for sources and listening and digging and watching every word and reading between the lines. By getting behind the scenes and into the back rooms and conference rooms, providing the stories behind the stories you get and the stories behind the stories you never get.

At MORE, we get media people to tell us things they'd never tell anyone else. And we get the very people who report, comment and advertise to write things about reporting, commentating and advertising they could never write anywhere else.

For example we've explored a family feud at the *Times* that may have been responsible for Daniel P. Moynihan becoming a U.S. Senator. We examined the power of a handful of editors at *Time* and *Newsweek* to create and destroy rock stars overnight. We interviewed the controversial *Los Angeles Times* reporter who said that journalists should "lie, cheat, steal or bribe to get their story." And MORE ran the Nora Ephron media column that *Esquire* killed and the profile of Rupert Murdoch that *New York* wouldn't run.

We've given the business to the news business, advertising, movies, publishing and the entire communications industry. And don't think they haven't started watching their words a little more closely now that they know someone else is.

So if you subscribe to the idea that someone should be watching the media like the media watches everyone else, subscribe to MORE.

I WANT TO KEEP THE MEDIA HONEST. SEND ME MorE.

Please enter my subscription immediately.

☐ $12 for 1 year (12 issues)
☐ $21 for 2 years (24 issues)
☐ $30 for 3 years (36 issues)

☐ Payment Enclosed ☐ Bill me

Name
Address
City State Zip
Signature

MorE THE MEDIA MAGAZINE P.O. Box 955 Farmingdale, New York 11735

MR57

Add $2.00 per year for outside U.S. and Canada. Please allow up to 4 to 6 weeks for delivery of first issue.

COMPUTER WRESTLING: The Program of Champions

by Lee Felsenstein

ROMtutorial ROMtutorial

K bytes: 1024 bytes (eight-bit units) of data. K usually stands for 1000, but in binary the closest even number is 1024. Quantities which are ultimately expressed using binary, like memory addresses, are commonly figured in batches of 1024.

TV Typewriter: One of the earliest personal computer devices, being a circuit board which converts data from a keyboard or a computer parallel data output to characters on a TV screen. The first TV Typewriter was designed by Don Lancaster and had circuitry to convert the video signal to high frequency form which could be accepted by the antenna input of the TV. At about that time the FCC cracked down and forbade these "RF modulators" except under stringent conditions.

TTL: Transistor-Transistor Logic. One type of internal design used in integrated circuits. Currently it's the major one in use, but things tend to change pretty rapidly.

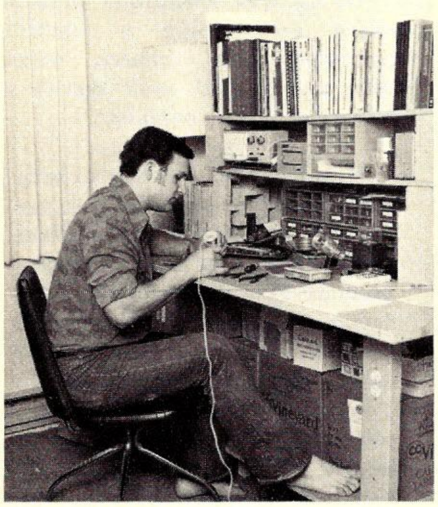
Remember the computerized Olympic scoreboards that couldn't handle Nadia's 100 percents in Montreal? At about the same time that those things were setting computers back to 1952 in the public mind, a young man of nineteen successfully designed, programmed, and operated a scoring system for the day-long wrestling tournament at his high school alma mater in California. His machine was an Altair 8800 with 40K bytes of memory and a cassette tape storage system. His scoreboard was a series of five TV sets fed by a "TV Typewriter." And his input device was a homebuilt punched card reader using paper clips for hole sensors!

> *His input device was a homebuilt punched card reader using paper clips for hole sensors!*

It worked (though not without a small hitch or two), and so well that those people at the tourney who didn't believe it was all being done in a back room with adding machines took it as just one of those things that you can expect these days.

John Kenneth Borders Jr. is a tall, sandy-haired guy of twenty-one who seems to have stepped out of the cast of *American Graffitti.* In fact, he went to high school in the central California town of Manteca, which is pretty much the area portrayed in that film. His modest, easy-going manner and sturdy build easily lead to the conclusion that Borders went in for athletics in high school, and this is indeed the case—he

The author and Mr. Borders discuss a point of fact.



Last minute pre-match adjustments are sometimes in order.

was on the wrestling team and played football. It's only when you talk computers with him that you discover that, while in elementary school, he built a little "simple calculator" using *marbles* for logic elements!

(Technical fact—a marble rolling on a see-saw type beam is a pretty good equivalent of an electronic "flip-flop" or "latch" circuit. You could use these kinds of mechanical elements to construct a digital computer or calculator, but don't write in and ask for plans.)

This achievement was followed by more: a relay computer that played *Tic-Tac-Toe* in his freshman year, a transistor calculator using "Nixie"-type readouts in his sophomore and junior years, and an integrated circuit version of the same calculator built in his senior year, along with an integrated-circuit TTL version of a four-bit computer built for use as a classroom demonstrator.

In elementary school he built a little "simple calculator" using marbles for logic elements.

By the time he graduated Manteca High in 1974, John Borders was well known for his computer activities. He went on to the Air Force Academy for a year studying electronic and microprocessor systems under the tutelage of a Captain Larsen, who also taught him about data acquisition and systems integration (putting boxes together). Illness forced him to leave the Academy in 1975, and he was one of

ROMtutorial ROMtutorial

Flip-flop: A device which can flip one way or flop the other way and be stable in either "state." Electronic "bi-stable" circuits like this can keep track of one bit of data and can be hooked together to form counters, etc.

Relay: An electrically operated mechanical switch. The whole telephone system used to consist of these, and is only gradually being replaced by a computer network. Relays operate through electromagnetic action.

Nixie-type readouts: Small neon-filled glass tubes in which the numbers 0 through 9 can be made to glow orange. The name is a trade mark of the Burroughs Corporation and stands for "Numerical Indicator, Experimental." Until recently, Nixies were the fastest and best numerical display.

Photographs by Garry L. Parrish

ALTAIR 8800 COMPUTER

So that's where you hide the half-nelson routine.

the first to put his money down for an Altair 8800 in that "Year One" of personal computing.

By 1976 Borders had his system together and running. Based on a MITS Altair 8800 with 8K bytes of MITS static memory and 16K bytes of Dynabyte dynamic memory, his box spoke with a Southwest Technical Products CT 1024 TV Typewriter for video display via a MITS 4PIO parallel interface board. A Singer 7102 one hundred-word-per-minute page printer (similar to a Flexowriter) was fed from a Processor Technology 3P + S serial/parallel interface card. He performed audio cassette storage of programs and data through a MITS ACR interface. MITS 8K BASIC (version 3.2) was running in the machine to allow him to program in the BASIC language, which is nearly conversational.

John drove a truck for two years to pay for the equipment and software.

John drove a truck for two years for his father to pay for the equipment and software, and he spent a year getting the parts together and debugged. Not being able to stand a thirty-two-character video line width, he modified his TV Typewriter to double the line length as soon as he had built it. He added scrolling and some other nice little features, too, such as automatic line clear.

California is actually several states, and the central valley area is best compared to any midwestern state. "These people are super-sports-oriented," explains Borders, "and a wrestling tournament is the event of the year. It's a community activity and everyone participates."

From his high school wrestling experience, Borders was keenly aware of the scoring and scheduling problem involved in running a wrestling tournament, with thirteen weight classes and a dual rank system of eliminations. As he explains:

> "It takes a lot of people to keep track of all the statistics that are coming in during a wrestling tournament. You need an instantaneous type of readout. The scores are coming in and the wrestlers are coming up and asking 'how am I doing, where am I standing, who am I going to wrestle next?' One person cannot handle all the data and all the people coming up and pestering him.
>
> "I foresaw that I could, by using a computer, provide instantaneous results so that at any moment I could tell any wrestler where he stood, exactly who he would be wrestling next, and at approximately what time he would be wrestling.
>
> "Also, at the end of the match, we were losing some very vital statistics because there was so much data that no one wanted to sit down and go through it all and compile it. I foresaw that if I could have all these statistics come into the computer they would be available and then I could have them output in some usable form at the end of the match to give to the wrestlers and

ROMtutorial ROMtutorial

Scrolling: A method of displaying data on a TV screen in which the display "rolls up" one line when a line has been filled. This makes it easier to read than the "paging" display in which the entire screenful disappears after the last line is written.

ASCII: Stands for American Standard Code for Information Interchange. A standard seven-bit code used to represent letters and numbers, as well as control functions like carriage-return, back space, etc.

the coaches so that they could see how their teams did as a whole or how individuals did in the tournament."

Having conceived of the system and what it could do, it remained for Borders to get permission to put it into effect. This wasn't simple. First he went to the coach, Larry Lathrop, who knew John's computer qualifications from his high school days. Larry thought it was a great idea, and thought John could pull it off, but he wasn't in charge, and had no power to make the decision.

Next stop was the man in charge, who was beset with worries about what would happen if the system crashed. John tried to explain how the system would be backed up on cassette tape at all times—indeed, the memory was big enough only for a limited amount of data, which would have to be read out to tape before the next batch of processing could begin. John had to calm a lot of fears and field a lot of excuses, but he prevailed largely because no one else wanted to handle the task. Eventually the official gave in.

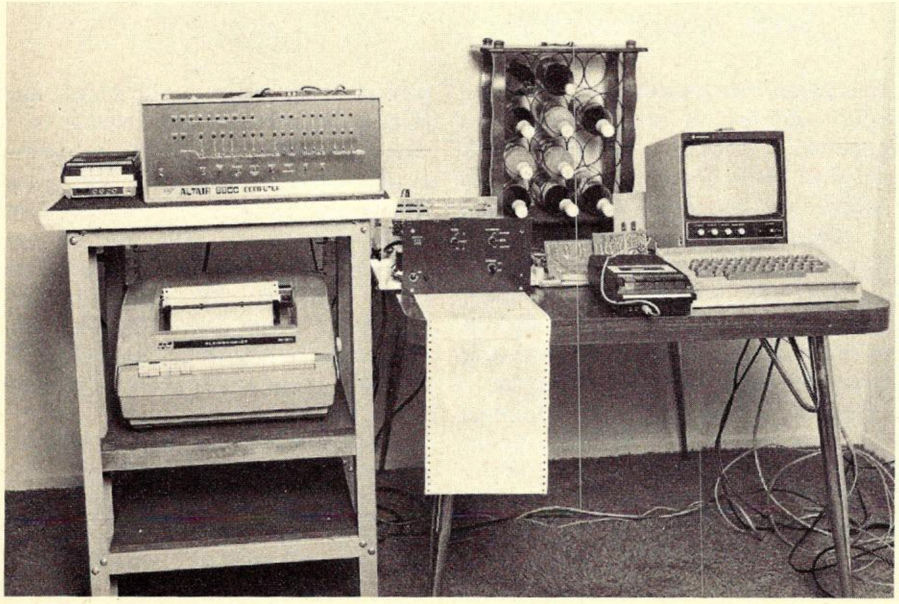
There are immense scoring and scheduling problems in running a wrestling tournament, with thirteen weight classes and a dual rank system of eliminations.

Next stop was the school district. They were concerned about safety. Obviously a computer meant lots of wires running all over the place and dangerous electrical circuitry. John had to assure them that the P.A. system would be handling much more power and would need even more wires than his computer. The objection arose, too, that the computer would take up the time of the people who would have to run the tournament. John had to explain patiently how the computer would be freeing lots of people from the scoring and scheduling tasks.

"It was all insane at the time, but I finally ploughed it through and got approval from everybody, mainly because I was the only person who wanted to accept the job. I guess that they didn't really give in, they just forfeited to me because there were no other contenders."

Fortunately, Borders didn't wait for approval to begin programming and planning. He had been working on the system all this time.

The current setup, including a Kleinschmidt 311 printer, replacing the original Singer, and a rack of post-match libation.



The system had to start operation as the contestants weighed in when they arrived in the morning. The name, school, age, weight class, and exact weight for each wrestler were entered along with a unique code number for that contestant. The data was entered into numeric and string arrays. After the tenth such array the computer prompted the operator to set up a tape file and output the data to tape.

After the weighing, these tapes were run through again to make up more tapes, this time organized by weight class, one cassette to a class. Then a new program was loaded to allow the scoring of the matches, which went on three at a time in each weight class. As the matches were completed, the wins, losses, and scores were digested by the system to determine the contestants in the next round. Losers of a match were matched with other losers to compete for a third and fourth place.

The standings were made available to the coaches on paper from the printer and on the TV screens to the audience. There was a method of interrupting the program to display messages on the TV screens as well (a lost child, the snack bar closing, etc.).

After the tournament, another program went in to compile the scores and statistics. Totals of scores, wins, and losses were added up and formatted by school, by wrestlers, and by weight classes. To do this, all of the score tapes of all the matches were run through once for each compilation.

"It took about fifty or sixty passes," Borders remembers. "I was running tapes all day. The recorder broke down about three quarters of the way through the tournament and we had to go out to Radio Shack and get a new one."

All tape files were in ASCII. Numeric values computed in the program (by adding scores, for instance), were in binary and had to be converted to ASCII by using the STR$ and CHR$ statements. The MID$ statement was used to output the ASCII strings to the cassette interface one ASCII character at a time.

John borrowed another 16K byte memory board for the development and operation of the program, giving him 40K bytes of memory. Of this

ROMtutorial ROMtutorial

POKE statement: A command in MITS BASIC which will cause the computer to change part of its own program in the manner commanded by the programmer. This "self-modifying" feature is one of the most potentially powerful capabilities of a computer. It is also one of the most risky, since it can lead to "crashes" when used by a clumsy programmer.

the BASIC interpreter used about 6.5K bytes, the actual program (written in BASIC) used another 4.5K bytes, and the arrays required 20 to 30 K bytes to avoid overflowing memory.

The hardware presented opportunities and requirements for ingenuity. First was the problem of data entry terminals. Borders couldn't afford to buy or build three full key-entry terminals, and was wary of running cables (expensive, noise prone, and ready to trip the unwary) around the gymnasium, so he developed his own punch-card system of taking the scoring information.

Borders didn't wait for approval; he had been working on the system all the time.

The scorekeepers at the mats had a supply of index cards with rows of punch positions Xeroxed along two edges. Borders taught them how to convert the numbers from the scores into BCD (binary coded decimal) numbers which could be represented by groups of four punches. They could then punch three decimal digits along each edge of the cards. These six digits corresponded to the code number of the wrestler, the weight class, and the match score.

The card reader Borders built consisted of a frame to hold the card in position while a row of twelve paper clips probed the punch areas to determine whether or not a hole was punched. Circuitry he had built sampled one four-bit group after another and converted the BCD digits to ASCII (eight bit) numeric codes. The circuit then terminated the three numbers with a carriage-return code. This was sent over a parallel interface to the computer, where the program thought that it was coming from a keyboard. The program was directed to accept inputs from the card reader by means of modifications made in the input driver section of BASIC through use of the POKE statement. After the input was accomplished, the program executed other POKE statements which returned the input driver to its keyboard condition. While the card reader was in operation, BASIC echoed the input to the TV Typewriter, so that a visual check of the incoming data was available.

Borders modified his TV Typewriter back to its original thirty-two-character line length for the tournament so that the audience could see the display clearly on the TV screens. He shipped the video signal around to the five TVs on a coaxial cable and had an ATV "Pix-e-Verter" at each TV to convert the video to modulated RF so as to go in through the TV's antenna terminals.

The computer and printer were tucked away in a coach's office just off the gym floor, and Borders set up his keyboard and card reader at the end of twenty-five-foot long cables out on the floor. Although he considered the electrical environment of the gym to be noisy and likely to inject false signals in these lines, he had no trouble with the setup.

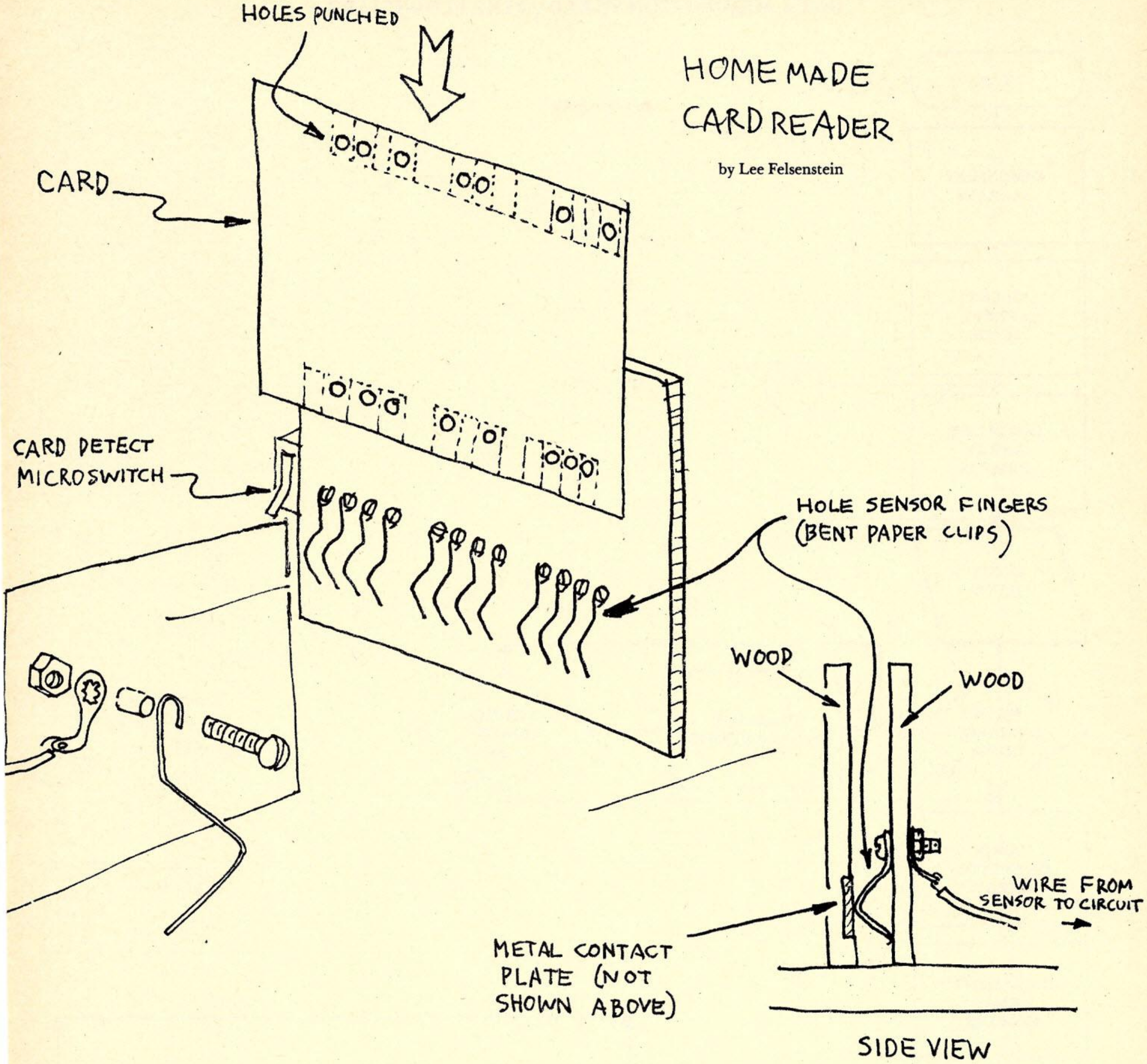
No data was lost or garbled, and the printed standings and results got raves from the coaches.

The tournament went smoothly, with only some minor machine malfunctions. The tape recorder breakdown mentioned previously was probably due to magnetized or dirty heads, Borders figures. The Singer printer broke down at one point, and John quickly traced the trouble to a charred resistor on one of the circuit cards. He had his tools and parts handy and replaced it. The machine resumed working, although no one ever found the cause of the burned resistor. Toward the end, the TV typewriter began giving trouble, possibly from overheating, but the TV display was not essential to the system's operation.

No data was lost or garbled, and the printed standings and results got raves from the coaches. To have the results in hand before departing was a new experience to most of them. While the general public at the meet took the

HOME MADE CARD READER

by Lee Felsenstein


HOLES PUNCHED
CARD
CARD DETECT MICROSWITCH
HOLE SENSOR FINGERS (BENT PAPER CLIPS)
WOOD
WOOD
WIRE FROM SENSOR TO CIRCUIT
METAL CONTACT PLATE (NOT SHOWN ABOVE)
SIDE VIEW

system pretty much for granted, the coaches came back later and asked Borders to do it again for the next meet.

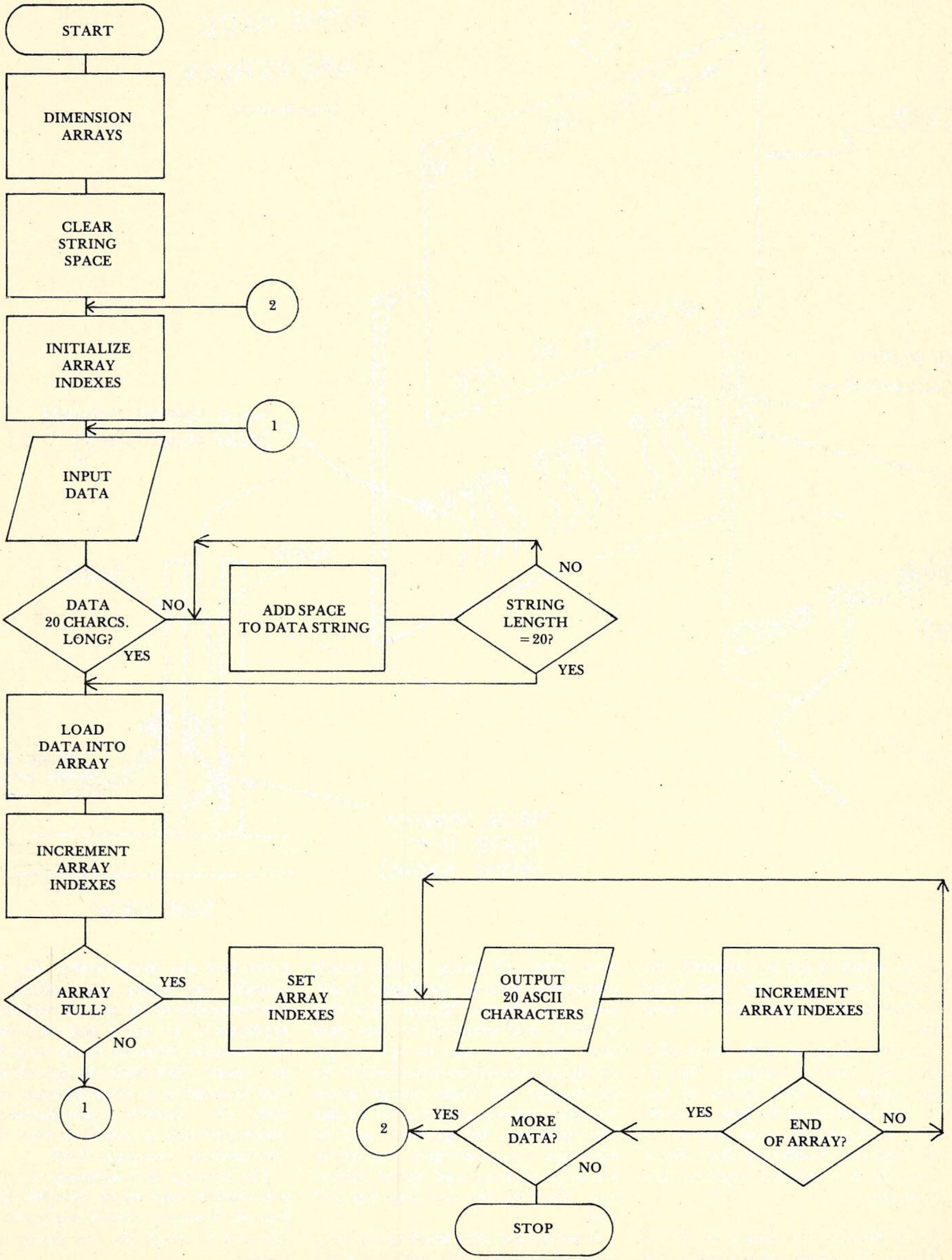
Unfortunately for them, he couldn't spare the time to organize the '77 tournament. He had moved to San Jose and gone to work, first at Lockheed Air Terminal, most recently at the Hayward, California Byte Shop, where he is a systems engineer and technician.

John Borders has shown us one of those numberless applications which were there all along, being poorly performed without intelligent hardware and software. No one asked him to do it, and certainly no one paid him—indeed, he had to fight to get the right to contribute his talents to his community. He knew enough about the technical requirements not to take his equipment for granted, and he allowed generous margins of safety in his programming and space allocation. Even so, the run was nip and tuck.

It seems that John Borders has done more than bring the personal computer into the sports arena. He has brought the values and outlook of sportsmanship to the area of computer application. To strive and overcome the obstacles inherent in the task for no reward other than the knowledge that he could do it—this contrasts well with the obsessive grandstanding which pervades so much of even the "professional" computer field.

The concept of computing as sport will require some more thought. But John K. Borders is clearly a pioneer in the area in which the two disciplines overlap. ▼

DATA ACQUISITION SUBROUTINE FLOWCHART


START
DIMENSION ARRAYS
CLEAR STRING SPACE
2
INITIALIZE ARRAY INDEXES
1
INPUT DATA
DATA 20 CHARCS. LONG?
NO
YES
ADD SPACE TO DATA STRING
STRING LENGTH = 20?
NO
YES
LOAD DATA INTO ARRAY
INCREMENT ARRAY INDEXES
ARRAY FULL?
YES
NO
1
SET ARRAY INDEXES
OUTPUT 20 ASCII CHARACTERS
INCREMENT ARRAY INDEXES
END OF ARRAY?
NO
YES
MORE DATA?
YES
2
NO
STOP

DATA ACQUISITION SUBROUTINE ANALYSIS

PROBLEM:

TO OBTAIN AND STORE DATA FROM EACH WRESTLER AS THEY WEIGH IN.
INPUT DEVICE IS DESIGNATED AS KEYBOARD.
OUTPUT DEVICE IS DESIGNATED AS TAPE UNIT.

INPUT:

DEVICE—KEYBOARD INPUT TO BASIC
DATA—
1. *WRESTLER'S NAME*
2. *WRESTLER'S SCHOOL*
3. *WRESTLER'S AGE*
4. *WRESTLER'S WEIGHT CLASS*
5. *WRESTLER'S ACTUAL WEIGHT*

FORMAT—ASCII STRING INPUTS TO BASIC

OUTPUT:

DEVICE—TAPE UNITS OUTPUT FROM BASIC
DATA—
1. *ARRAY FILLED WITH INPUT DATA*
2. *10 DATA INPUT RECORDS PER LOOP*

FORMAT—ASCII STRING OUTPUTS FROM ARRAY IN BASIC. (ONE CHARACTER AT A TIME.)

STATISTICAL SUBROUTINE ANALYSIS

PROBLEM:

TO OBTAIN DATA FROM MAGNETIC TAPE, COMPILE STATISTICS AND OUTPUT STATISTICS TO PRINTER.

INPUT:

DEVICE—TAPE UNITS INPUT TO BASIC
DATA—
1. *WRESTLER'S NAME*
2. *WRESTLER'S SCHOOL*
3. *WRESTLER'S AGE*
4. *WRESTLER'S WEIGHT CLASS*
5. *WRESTLER'S ACTUAL WEIGHT*
6. *NUMBER OF MATCHES WRESTLED*
7. *NUMBER OF MATCHES WON*
8. *NUMBER OF MATCHES LOST*
9. *PLACING IN WEIGHT CLASS*
10. *TOTAL POINTS IN TOURNAMENT*
11. *MATCH TIMES (TIME OF DAY)*
12. *INJURIES (YES OR NO)*

FORMAT—ASCII STRING INPUTS TO BASIC

OUTPUT:

DEVICE—PRINTER OUTPUT FROM BASIC
DATA—

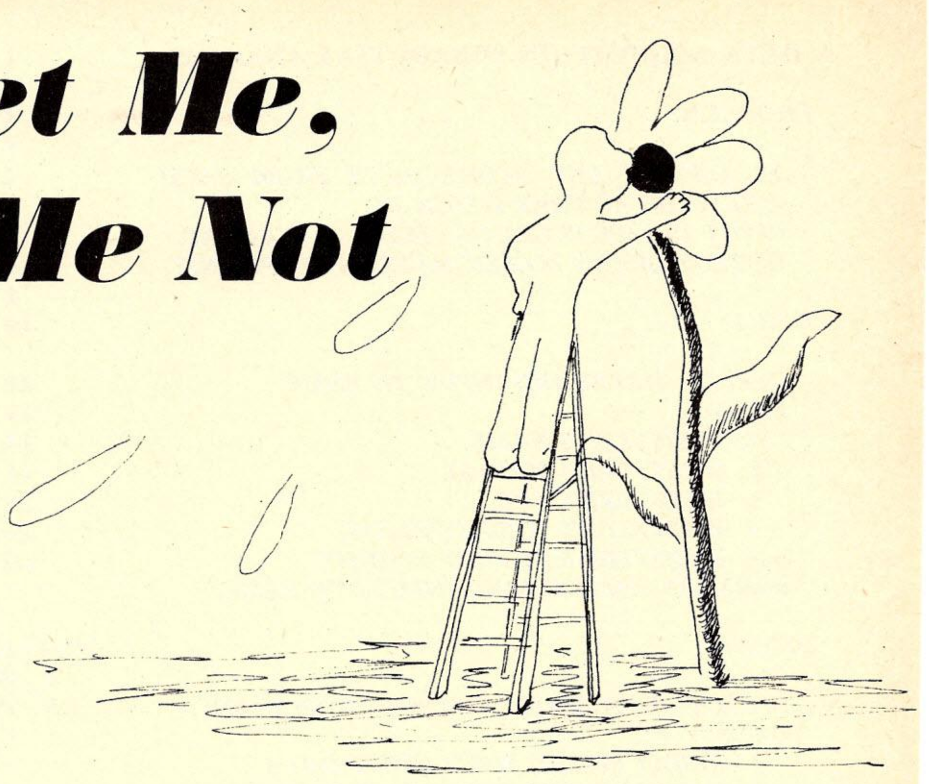
PASS #1—BY SCHOOL
- *A. WRESTLER DATA = DETAIL LINE*
- *B. TOTALS FOR SCHOOL*
  1. *NUMBER OF MATCHES WRESTLED*
  2. *NUMBER OF MATCHES WON*
  3. *NUMBER OF MATCHES LOST*
  4. *NUMBER OF FIRST PLACES*
  5. *NUMBER OF SECOND PLACES*
  6. *NUMBER OF THIRD PLACES*
  7. *NUMBER OF FOURTH PLACES*
  8. *TOTAL NUMBER OF POINTS*
  9. *TOTAL NUMBER OF INJURIES*
  10. *NUMBER OF WRESTLERS PER WEIGHT CLASS*
  11. *NUMBER OF WRESTLERS PER AGE*
  12. *AVERAGE AGE*
  13. *AVERAGE ACTUAL WEIGHT*
  14. *PERCENTAGE OF MATCHES WON*
  15. *PERCENTAGE OF MATCHES LOST*
  16. *PLACING IN TOURNAMENT*
  17. *TIME OF DAY GRAPH*

PASS #2—BY WEIGHT CLASS
- *A. WRESTLER DATA = DETAIL LINE*
- *B. TOTALS FOR WEIGHT CLASS*
  1. *NUMBER OF MATCHES WRESTLED*
  2. *NUMBER OF MATCHES WON*
  3. *NUMBER OF MATCHES LOST*
  4. *NAME OF FIRST PLACE WRESTLER*
  5. *NAME OF SECOND PLACE WRESTLER*
  6. *NAME OF THIRD PLACE WRESTLER*
  7. *NAME OF FOURTH PLACE WRESTLER*
  8. *TOTAL NUMBER OF POINTS*
  9. *TOTAL NUMBER OF INJURIES*
  10. *AVERAGE ACTUAL WEIGHT*
  11. *AVERAGE ACTUAL AGE*
  12. *NUMBER OF WRESTLERS PER SCHOOL*
  13. *NUMBER OF WRESTLERS PER AGE*
  14. *NUMBER OF WRESTLERS IN WEIGHT CLASS*
  15. *TIME OF DAY GRAPH*

PASS #3—BY INDIVIDUAL WRESTLER
- *A. WRESTLER DATA = DETAIL LINE*
- *B. TOTALS FOR TOURNAMENT*
  1. *NUMBER OF MATCHES WRESTLED*
  2. *NUMBER OF MATCHES WON*
  3. *NUMBER OF MATCHES LOST*
  4. *NUMBER OF WRESTLERS PER WEIGHT CLASS*
  5. *NUMBER OF WRESTLERS PER AGE*
  6. *NUMBER OF WRESTLERS PER SCHOOL*
  7. *AVERAGE AGE*
  8. *AVERAGE ACTUAL WEIGHT*
  9. *TOTAL NUMBER OF POINTS*
  10. *TOTAL NUMBER OF INJURIES*
  11. *SCHOOL PLACING*
  12. *WRESTLER PLACING*
  13. *TOTAL NUMBER OF WRESTLERS*
  14. *NUMBER OF WRESTLERS PER WEIGHT CLASS*
  15. *TIME OF DAY GRAPH*

FORMAT—ASCII STRING OUTPUTS FROM BASIC

Forget Me, Forget Me Not



by Avery Johnson

"The evil that men do lives after them.
The good is oft interred with their
bones."

With these words, and a few hundred others, Mark Antony bade farewell to Julius Caesar—and the canny bard had put his finger on a problem of human nature that is with us today in even more virulent forms than he could have imagined.

How so? Because our ways of remembering what people do are on the one hand so much more copious and precise, and on the other hand so much further removed from the contexts of their happening. What can we do about it? Why does memory seem such a fragile and mutable thing in our own psyches, and then such a threatening monster poised to bring us down when it is lodged in data banks beyond our reach?

Data banks are recorded fingers pointing at events, but the events themselves cannot be reconstructed.

Let's go to a personal level for a moment. Some fragment of information gets back to us that someone told to someone else about something we did, and we stand there horrified at the misinterpretation of the same "facts" that we remember.

"That was taken out of context!" we want to shout, and that is exactly the point.

Yes, it was. It's something we are coping with all the time, but we only become aware of dangers when a damaging finger is pointing at *us*. As a language-using and symbol-manipulating species we are forever involved in having to carry along with our pronouncements a sufficient amount of contextual matrix to provide meaning for the facts embedded in it. If we don't, we will be misunderstood. No other species has the gift or the problem.

Point your finger at something distant that catches your attention and the person with you will look in the direction that you point. The chances are also very good that whatever it was that caught your eye will catch other human eyes as well, because they share approximately your habits of looking and of interacting with the world. But don't bother to try to point something out to a dog or a chimpanzee or a dolphin, because any other creature is probably just going to focus its attention on the end of your finger. Other creatures simply don't catch the notion of "intention," of reference to an experience they might have or have had in a time and place different from the here and now.

Thus it is with data banks and other forms of "photographic memories:" they are recorded fingers pointing at events, but the events themselves cannot be reconstructed from the necessarily finite collection of relata on file. The *relations* that characterized them as events to be experienced have been lost.

Well, what then? Are we to be inevitably and forever saddled with this problem? Are we never going to be able to trust our ability to reconstruct the essence of our personal histories from the facts we can record? No, I don't think it's as bad as all that. There are some steps we must take, some ambiguities we must come to recognize, and some ultimate limitations we must accept, but we can certainly advance beyond our present impasse. In the future we need not be so paranoid about data banks nor so secretive with our records—if we adopt now a doubter's credo for our own avenues of perception.

Let's digress again for a moment. I introduce you to a friend of mine and ask you to talk to him for a while, then to come and see me and tell me what you've learned. The first thing you say to me later is: "Hey, he's crazy; I mean out of it!"

"Why do you think that?"

Illustrated by Sue Bass

"Well, I can't make sense out of anything he says."

"OK. Some others agree with you in that he has been officially labelled as schizophrenic, but what in particular can you say confused you?"

"Let's see. Most of the things he said seemed as though they ought to be meaningful by themselves, but somehow they didn't relate to each other in any way that I could fathom."

How about the requirement that any data entry about a person carry with it the time, place, and "surrounding circumstances?"

"Do you think they made sense to him?"

"Probably, but I couldn't share that space with him. I had the feeling that sometimes he was trying to put me on, but that at other times he was trying to get something across that was real to him."

"Sounds like your own paranoia was being tapped into, but you've essentially described the experience of talking to him. Do you think that if you talked to him for long enough, you could begin to share his world and find his statements meaningful?"

"Perhaps, but what is 'long enough?'"

"I don't know. It might depend on you; it might depend on him. There's no way to tell whether you could ever 'tune in.'"

I have talked with psychiatrists and others who have worked with schizophrenics and they generally agree that the initial stages are difficult, but that given enough time they can, in fact, come to share the contextual relationships of the person they are talking to and that, thereafter, the world described by that person no longer appears crazy. The relata and their relations are now appropriate together. The schizophrenia momentarily disappears.

Aside from sharing a long enough "here and now" to allow us to pick up on the frame of reference of my friend, what else might be involved in learning how to make sense out of what at first looked like a nonsensical word salad? That's hard to say. It's a process whose elements are certainly not of a measurable, quantitative form, but that does not rule them out as candidates for modelling by computer.

Much of the process is structured into the elegant and mysterious ways in which we use language. It is well beyond syntactical analysis, coming more into the realm of the poets. I would like to leave this question hanging for a future article in *ROM* that will deal more directly with the problems of a logic of relations and of some of the ways in which our language both hides and reveals those relations. For the moment the relevant issue is one of memory: memory of what and how do we use it wisely?

Nowadays we seem to have the greatest respect for storage and retrieval systems that mimic a combination of office copy machines and filing cabinets. Time was, before the invention of carbon paper (in 1925 specifically for use in lawyers' offices), that copies had to be made laboriously by hand and scriveners were hired everywhere simply for that job. Charles Dickens's scenes come to mind: miserable people with cold fingers and dulled minds. Carbon paper at one leap allowed, at a single typing, a number of identical copies of documents, which had to be identical in order to have any binding contractual value at all.

Office copiers are our present scourge that grew out of our copymania, but they serve in no way to make the information on the copy more meaningful, relevant, retrievable, or storable—just a whole lot more of it everywhere. Imagine what havoc the disinvention of the copiers could wreak! Maybe it's unfortunate that invented devices are not like newly-mutated species; they create their own ecological niches and a vast appetite among their users for more and "better," but do not succumb to any laws of extinction. I am not proposing a return to the days of the scriveners, but rather a rethinking of what it is we want to remember and a respect for our responsibilities in working out an adequate approach.

When we hear something bad about someone we like, we find ourselves asking that the person be given "the benefit of the doubt." We recognize that there is ambiguity in any telling and that a resolution of the ambiguity could change the effect of the telling in any of a number of ways. A first rule might therefore always be to recite information about someone aloud in that person's presence before acting on it. Such is the intention, in law, of allowing the accused to confront the accusers, but in actual cases, like Credit Bureau listings, a satisfactory recitation may be impractical.

Next best might be the requirement that any data entry about a person carry with it the time, place, and "surrounding circumstances," as well as the name of the person with whom the entry originated. Difficult to enact or enforce? Yes, but my purpose here is not to generate new restrictive laws, but to start a formulation of what a "sufficiently" guarded attitude on our part might be, to keep us from falling into the trap of assigning meaning to context-free statements.

Much of what we want to know about another person does not fall into the category of fact so much as into that of opinion: an assessment by others of, say, what it is like to enter into a relationship with that person. We want an inductive grasp of A's attitudes and abilities without having to take into account the particular and limited actual experiences of B in making the assessment. It's a tough one: not so bad if we trust B, but if the information is stored in a data bank, who was B? In what context did B know A? When was it?

If and when we can invent an adequate relational calculus (as I said, that's for another article) we may not always be able to assure ourselves that the relational qualities of a data bank entry will be maintained, but we will at least have a handle on the relative level of ambiguity in the entry and therefore a feel for whether we should discount it or not.

The day may come when none of us will mind the fact that information about us, like our yearly income or our national origin or even a record of imprisonment, is available on a computer to the public at large. By then it will be everywhere accepted that discrete chunks of such indiscreet data are truly meaningless in the best (worst?) sense of the term. ▼

Herman Fuller is browsing through the evening's news printout when he comes across his own obituary. He reads with a numbness around his ears the details of his life that have been stored in the memory of the *Life and Times Network News* computers and information retrieval system:

> FULLER, HERMAN T. Author of the famous Computerized Outlaw Prevention Program (COPP) that has practically perfected prediction of anti-social, criminal events. Noted lecturer and political advisor, president for many years of Computer Applications for Social Problems Institute (CASPI). Eccentric, famed for his hobby of raising biocomputer organisms, actually combinations of bacteria and ssmpu's which could perform circus tricks on a microscopic scale. Survived by a daughter, Lynne, well-known printout cover model for National Fashion Syndicate.

"What journalism has come to," Fuller thinks. "They give my hobby more space than my professional accomplishments. And imagine them putting me in the past tense already. Retired, yes, but not exactly cold meat yet, the bastards."

* * *

Fuller takes a walk to cool his anger about the obituary and decide how to respond. Terrible, that circus business. Some of his critics say his recombinant work with bacteria and ssmpu's will outlive COPP, but he doesn't believe it. *Life and Times* has simply missed the point, the main point of his life, after all: the crime prevention program which has made it possible once again to enjoy the streets and alleys of western world cities, to inhabit a nuclear-powered and highly technologized planet without fear of sabotage and hijackings, kidnappings and terrorism. He's simply going to have to give the *Life and Times* computer-obit editor a readout that will kindle his batch.

He'd had plenty of dealings with *Life and Times* in the past, and they knew Fuller very well. They had been opposed to the government's adopting COPP in the beginning, and Fuller had composed many an angry holotape to them, only to find his arguments so mangled in the editing that they were sometimes turned on their head. He had a score or two to settle with *Life and Times* as it was, and publishing his obituary was simply unforgivable. For this, coming out of retirement was imperative. He'd show them who was dead.

* * *

Fuller punches the *Life and Times* contact code on his holovision and leans back in his chair, fuming. After the usual wrestling with pseudo-images on holotape telling him that Ed Billings is in conference, Fuller finally programs his way into the central office. Billings' simulation appears in the holovision console, but Fuller knows a pre-packaged image (known as a "dodge" in the trade) when he sees it and erases the fake with his concentrator. This brings results. Billings appears at last—balder, dumpier, altogether more tired looking than his corporate pseudo-image so carefully manufactured by the community relations engineers in his megacorporation. Fuller checks his own image on a small monitoring screen and modulates the red tones to make himself look fiercer to Billings, even angrier than he is.

"What's the matter, Ed?" Fuller cackles. "You look like you're seeing a ghost."

"My image ID system says I'm talking to Herman T. Fuller," Billings says, "but I'm pretty sure we killed you off this morning. Did we make a mistake?"

"Not really. I'm just calling in from heaven to let you know what things are like up here."

"Cozy, I hope," Billings says.

"I want a correction, you idiot," Fuller booms. "Half my records will probably be destroyed as a result of your blunder, and I'll be up against a lot of hassle and confusion just when I want most to get away from all that. Unless I get some satisfaction, I believe I'll sue."

"Believe me, we're very sorry about this, Dr. Fuller." Billings up-modulates the gray to make himself appear suitably abject. "But I think we can explain."

"I doubt it, but why don't you try anyway."

"Well, according to our records, 750 calls were made to your home without any response. Normally that means the source is out of service entirely, dead, at least from an information-gathering standpoint, and on the 751st unanswered call, the relay is tripped for obituary activation."

"Surely you didn't try to call me 751 times in the last two weeks?"

"No, but, ah, other people have." Billings grins sheepishly.

"In the first place, you're not supposed to have that information. Privacy Protection Act," Fuller says.

"True, but in the case of celebrities. . . . You know, we hate to be scooped."

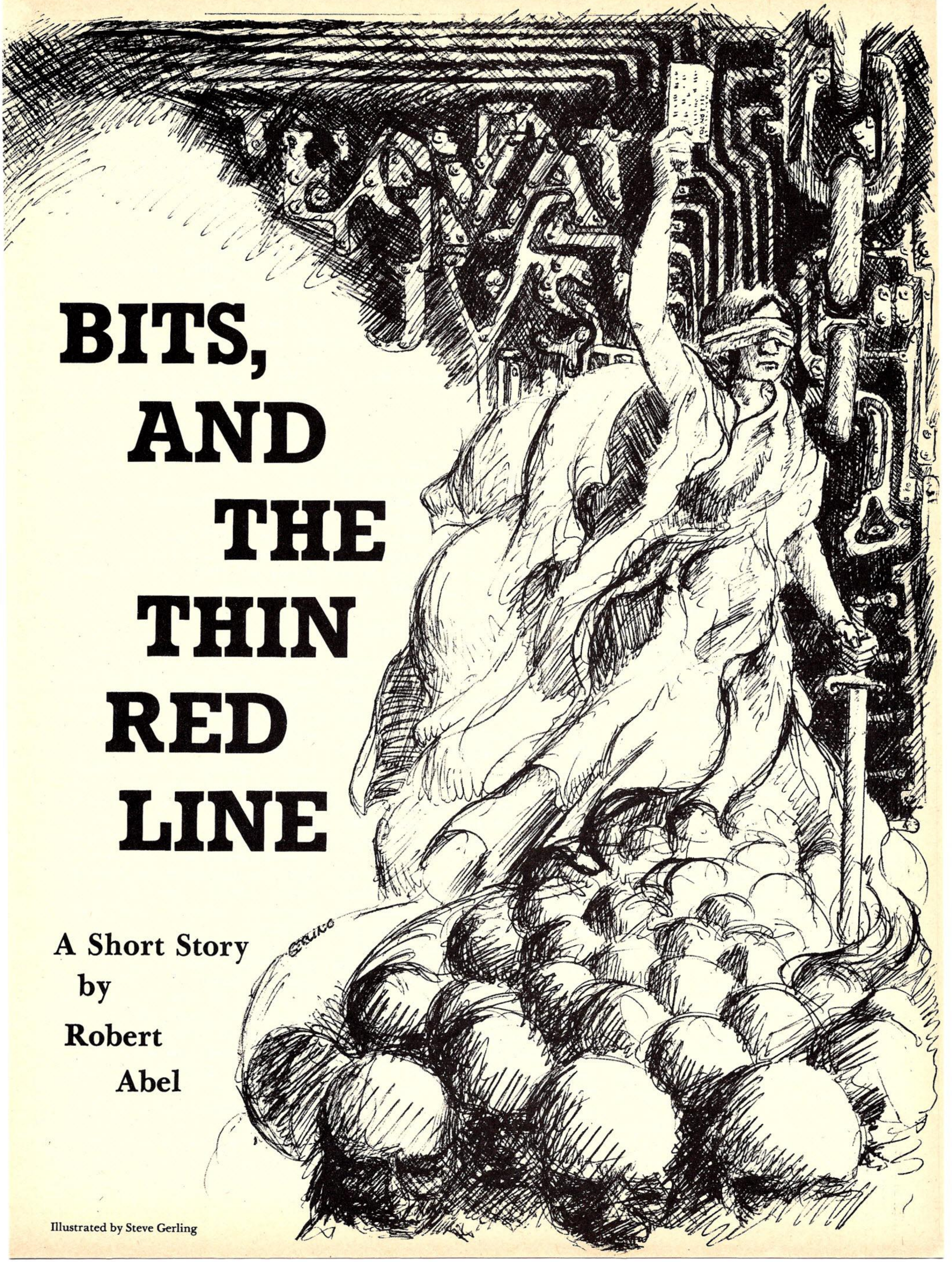
"Would you like me to kill myself or anything, just to keep you ahead of the competition?"

"No, of course not, but. . . ."

"In the second place," Fuller interrupts, "I didn't answer my holovision because I'm in retirement. Can you imagine what my retirement would be like if I answered every one of those

BITS, AND THE THIN RED LINE

A Short Story
by
Robert
Abel



Illustrated by Steve Gerling

751 calls? Can you? Do me a favor, Billings, and let the world know I'm retired, and that I just don't want to be bothered."

"It wouldn't be as bad as you think," Billings says. "313 of those calls were from the same source."

"Rewriting COPP is nonsense. That program's good for fifty years anyway."

"You nosey bastards," Fuller says coldly. "I suppose you can also tell me who they are and what they want?"

"It's CASPI, and I believe they want you to do some consulting, or possibly to rewrite the COPP program. But of course I couldn't know for sure."

"Rewriting COPP is nonsense," Fuller says. "That program's good for fifty years anyway. It's a monument."

"So you're coming out of retirement, are you? Can we run that?"

"Hell no!" Fuller twists the red-tone knob full and modifies to modulate in some demonic yellow flashes. "I want a retraction of that obituary, a statement of my retirement, and I want you to get rid of that sensational nonsense about my bacterial circus. When I do die, you can at least lend me the honor of letting the COPP program stand or fall on its own merits. That's all I have to say."

Fuller beams out, but not before programming in an interference message he learned in the Compuweapons Corps. Billings' office would be rattling with static for an hour!

* * *

Now Fuller programs in CASPI to find out why they have tried to reach him 313 times in the last two weeks. "Blasted short retirement," he groans. "Out of service for two weeks and already they're trying to wreck my masterpiece."

Harry Sanders leaps out of his chair when Fuller's image appears in the holovision.

"But we read you were. . . ."

"Dead? Harry, we social engineers never die. We just get our names in the news printout. What the devil's going on?"

"Dr. Drysdale will be happy to know you're *alive.* We were all so shocked."

"Tell me about it," Fuller says.

"I'll switch you right in."

The holovision image blips, blinks; Fuller catches a glimpse of his former colleague struggling into a pair of pants. Jogging shorts and tennis shoes are on the floor. As soon as Drysdale sees Fuller's image in the console, she staggers back, zipping and buckling. Then she smiles.

"Hey, Herman," she says, throwing open her arms. "Are you going to sue *Life and Times?*"

"I just love those polka-dot shorts," Fuller says.

* * *

Drysdale reports that her attempts to contact Fuller are motivated by "some troublesome current events. It's come to our attention that about half the artists in the country are in prison as a result of COPP arrests, for example. Now I'm not saying there's anything wrong with COPP, don't misunderstand me."

"Then just what is it you expect me to understand?" Fuller asks.

"Now, take it easy, Herm. Of course you remember the old days when COPP first came to national attention. It was when you made a prediction that a crime would occur at the Smallville First National Bank at 1:35 Friday, March 12."

"Right."

"And we had the holovision cameras and the police and the people from *Life and Times* already on the scene."

"Right."

"And at 1:35, in walk these two poor, unsuspecting bank robbers, masks on and guns drawn, completely oblivious to the fact they were about to become the most famous crooks in history. They had just put their pistols in the tellers' faces when all the holovision beams came on and the microphones and cameras popped out from under the coats of the newspeople. 'How does it feel to be the first criminals ever apprehended in advance of the crime?' That kind of stuff."

"So what? That's old hat."

"Yes. The point is that these crooks, Ed Duffy and Maggie Kirkwood, made their intentions pretty plain when they appeared with guns and masks. *They* were arrested in the actual attempt to commit a crime."

"And they weren't the last."

"Of course not," Drysdale exclaims. "They were the first of thousands to be caught in the act. The program was so good that after a few years anybody arrested on COPP was automatically sentenced, the guilt was so obvious. Let's not forget that."

"Who's forgetting? I still don't know what you're talking about."

"Can you come down to CASPI right away? I have some holotapes of a few courtroom sessions I'd like you to look at."

"Hey. Don't you know about the Privacy Act? Courtrooms can't be taped."

Drysdale waves the remark away. "Oh come on. It's in the interest of science. Can you get here today or not?"

"I can, but I'll charge a consulting fee."

"Great, and I'll throw in a dinner."

"I still think you'd better read the Privacy Act," Fuller says. "*Somebody* ought to anyway."

* * *

Drysdale plays a number of holotapes of court proceedings involving COPP arrests. Fuller is a little bored by them. He watches calmly as a prosecutor plies his trade:

"Your Honor, the defendant displays a high risk for social deviancy and was found to be in a crime-precipitating situation. His attitude profile reflects a very limited array of perceptual tones—anger, disgust, jealousy, frustration—and the Data Base Card displays the full scarlet line which indicates tendencies toward high-risk behavior and crime potential."

"Any redeeming factors?" the judge asks. "What are the superego and will-power quotients?"

"1.2 and 1.4, Your Honor, which are borderline psychopathic."

"I know the definitions, Mr. Murdoch, thank you. What are the love and respect gradients? You didn't mention these."

"They weren't worth mentioning, Your Honor."

"Just stick to the facts, Murdoch."

"Yes, sir. They are 2.01 and 1.99 respectively."

"Would you consider these average scores?"

"Below average, Your Honor."

"I see. You have very high standards, Mr. Murdoch."

"No, sir. I'm just from the old school."

"Do you expect everyone to have 4.0 in love and respect?"

"No, Your Honor. But a 3.2 isn't bad."

* * *

"I don't see any problem with these cases," Fuller says when the holovision is switched off. "As far as I can tell, they follow the tried and true CASPI formula, HRP plus PC equals BT: High Risk Person in a Precipitating Circumstance means Big Trouble."

Suppose a deviant got hold of a kilo of plutonium. Then how do you punish some kook for destroying Minneapolis?

Drysdale pulls away from Fuller, seeming to be a little shocked. She shakes her mane of grey-white hair and peers at him sternly over her eagle nose. "The difference is that these people haven't done anything. And that's a big difference. They aren't pulling pistols, they're just out walking the streets, you old fool."

The words sting. Fuller closes his eyes a moment. "No fool like a dead fool, I guess."

"I'm sorry, Herman. But you seem to be losing perspective. These people are going to jail without even attempting any crime."

"Sooner or later they'd be in jail. Those were disgusting profiles. Every one was clearly HRP."

"HRP is a potential, not a fact."

"Maybe the sun won't come up tomorrow," Fuller says. "It has that potential, too."

"Now don't play God," Drysdale says. "I know better."

"I'm not playing God, damn it. I'm just being scientific. COPP is very well-established as a predictor of criminal events. That's indisputable. And it is also true that we must have a crime prevention system, not just a crime response mechanism. You know very well what would happen if a deviant got hold of a kilo of plutonium. Then how do you punish some kook for destroying Minneapolis? At this point, our only choice seems to be either to live in terror or to trust COPP to keep the peace. I'm sorry, Drysdale, but I think you want to sell our civilization down the tubes for an anachronistic nicety of civil liberties. Is that it?"

"Of course I wouldn't put it that way." Drysdale reaches for the holovision console and taps in a code. "Now it's my turn to play dirty," she says. "You'll remember this little event from the documentary I am about to show."

The holovision sparkles to life. A huge crowd of demonstrators is pressing against a wire mesh fence surrounding a nuclear power plant.

"Remember?" Drysdale asks. "The Washington, D.C. Nuclear Power Consortium. Back in 1999. 460,000 demonstrators, all angry with two recent loss-of-coolant accidents at the site."

"You *are* playing dirty," Fuller says. "Shut that damn thing off. Neither of those accidents was associated with any unacceptable radiation levels."

"But unacceptable to the demonstrators."

"HRPs, all of them."

"By current definitions, yes," Drysdale says.

"Shut it off, I don't want to see it."

"This was the third loss-of-coolant accident, rather poorly timed. Or maybe not?"

"Looks like you'd better come along with me," the officer says. "You've got a full red line."

Suddenly a super-heated steam cloud mushrooms up from the plant and tumbles into the demonstration. There is a terrible sizzling as the crowd is instantly transformed into little bits of dust and gas.

"Turn it off!"

"All right." Drysdale flicks the switch.

In Fuller's mind, the dust of the demonstrators settles down slowly, like invisible ash. "Now let me just tell you something," he growls. "As far as I am concerned, that little horror is ancient history. And from my point of view that accident was tragic in more ways than one. It didn't just kill people. It eliminated virtually all of the non-violent opposition to nuclear power. And that meant the only opposition left was either political, a few old guard Democrats, or subversive. And when you have subversives and nuclear power, you must have COPP! That's how I see it."

"Let's go to dinner," Drysdale says.

"You're not really hungry after that, are you?" Fuller points to the holovision.

"Starved," Drysdale says.

Now it is Fuller's turn to play shocked. "If I wasn't retired, I'd be worried about your HRP," he says.

* * *

Dinner is a little too chummy, Fuller thinks. Something is out of whack. Drysdale drinks too much, which is unusual for a person normally so politic, even canny. She is garrulous and apparently cheerful, but Fuller does not let his guard down. "She's after something," he thinks. "I'd better avoid a precipitating circumstance."

They argue about COPP, but Drysdale does not convince Fuller that the system has carried "prevention" too far, into the realm of "aggression against differences." She has been pressing the point (and also Fuller's thigh—an unheard of liberty in the old days) that applying COPP so rigorously denies any possibility of choice or free will. You're either on the program or you're not, she claims.

"Maybe it would be worth it," she insists, "to live with a little less security, with a little more fear if you like, and have a little more personal freedom. We're not machines, are we?"

Fuller takes Drysdale's hand from his thigh and places it on the table. "No," he says. "But then we're not criminals, either."

* * *

Outside on the street, while they are waiting for a taxi, Drysdale's drunken chatter draws the attention of a Personality Inventory Service Officer. He strolls over, tips his hat.

"Let's see your Data Base Cards, please," the officer says.

"It's all right, officer, we're from CASPI," Fuller says. "Dr. Drysdale and I have just been celebrating."

"That's fine," the officer says. "All the same, I'll have a look if you don't mind."

"Certainly," Drysdale says, handing over her card. "But you ought to know Dr. Fuller, after all. His work was the foundation for the Personality Service, and made your job possible."

"Then I'm very obliged," the officer says. He punches Drysdale's card into the analysis box and after a moment hands it back. "Thank you, doctor. And now, sir, your card, please."

Fuller's card slides into the box, and does not emerge. The box clicks ferociously.

"Looks like you'd better come along with me," the officer says. "You've got a full red line."

"Oh, nonsense. We really have work to do," Fuller says.

"But you're under arrest."

"What the devil for?" Drysdale shouts.

"Possession of a deceased person's identification."

Fuller swears. "Officer, I can explain."

"Not to me, I'm afraid." The officer takes Fuller's arm and propels him down the street as if he were a doll.

* * *

It takes two days, but finally Fuller is released on his own recognizance. Image-identification systems corroborate Fuller's claim to be alive and well, and the *Life and Times* retraction of the obituary is also produced as evidence. Drysdale testifies to their long professional association.

All the same, Fuller emerges from prison in a furious mood. He conceals the mood well, and is even suave in his last interview with the judge, who comments on the irony of Fuller's arrest by a system he himself had perfected.

"I believe it's the only mistake I ever heard of," the judge says. "How odd that it should involve you."

"Isn't history curious, though?" Fuller smiles. He is anxious to see Drysdale again.

* * *

"You're right," Fuller says. "COPP needs to be revised. I'm coming out of retirement. Can you get me some office space with access terminals to government computers?"

"Of course," Drysdale says. "Prison seems to have changed you."

"Present company excepted, I never met more interesting people," Fuller says.

* * *

A meeting is arranged with the Personality Service. It is strictly confidential. Fuller and Drysdale make their case for revising COPP to provide for the possibility of personal freedom, the expression of creative differences.

> *"The possibility of a free, humane choice exists until the last moment when the criminal act is initiated."*

"The possibility of a free, humane choice exists until the last moment when the criminal act is initiated," Drysdale says.

Fuller gives his full support. "We shouldn't be too proud of ourselves," he says. "I've learned that myself. We are not gods, and each person deserves the right to make a mistake. Total prevention of crime deprives us of practicing the very human art of forgiveness, don't you think?"

* * *

The Chief of the Personality Inventory Service replies: "I find these arguments shocking, and indicative of a callous disregard for the safety of our civilization and the planet as a whole. The request to dismantle or modify the advance crime prevention program is denied. Furthermore, both of you will be held for personality testing, observation, and questioning."

* * *

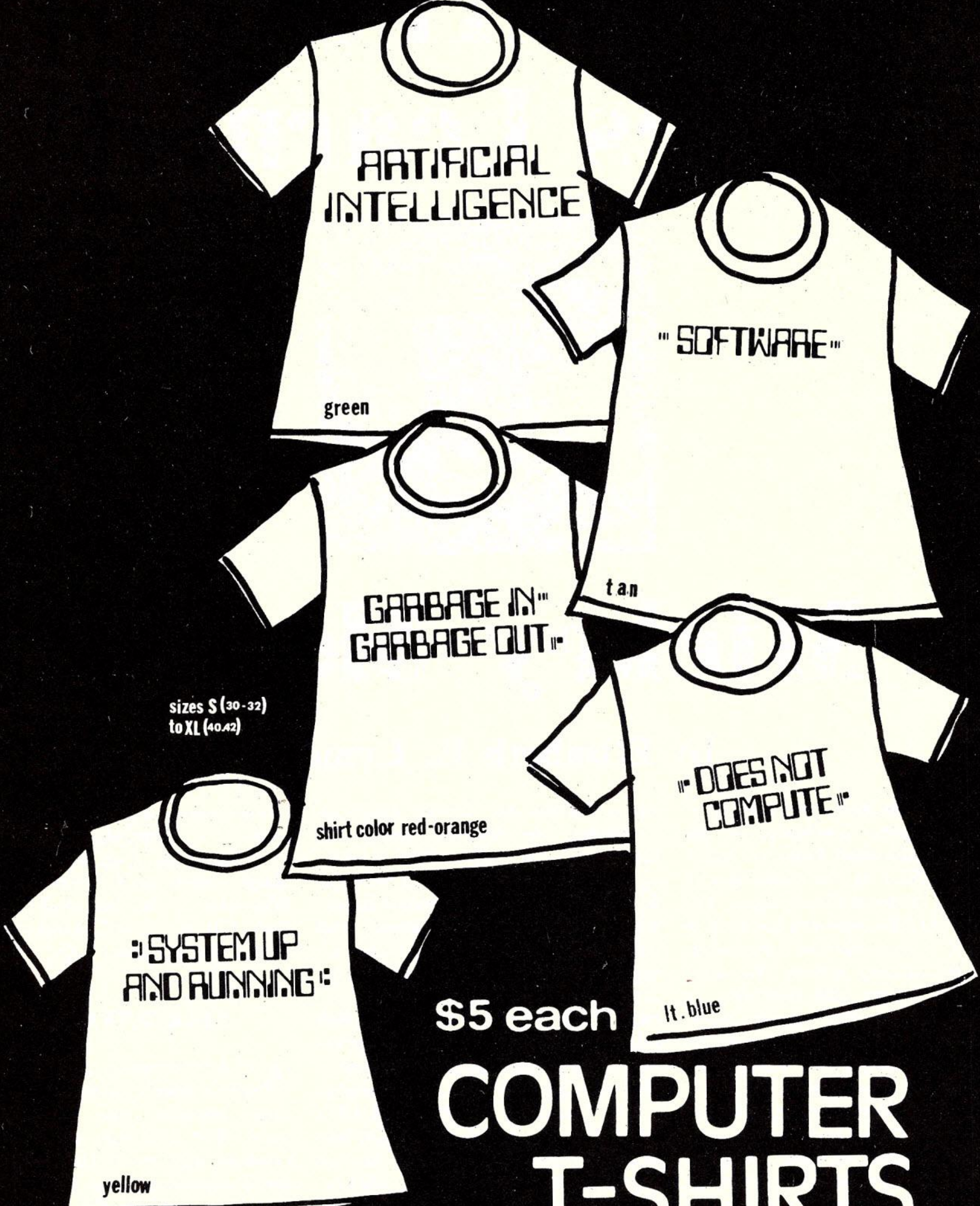
In his cell, Fuller is allowed the privilege of reading an edited version of the evening news printout. He reads it over and over again to keep from wondering what has happened to Drysdale, what they may be doing to her. He rereads an article on a message that arrived from distant space, which carried the atomic number to six hundred places. The article said, additionally, that each of the numbers appeared to have come from a separate planetary source.

This amazing message stream indicated a complex of planets integrated into a vast, communicating network of beings that made western world civilization seem to be as intelligent as a retarded grapefruit. To those who believed earth's creatures to be of supreme intelligence, it was a dismal blow. But to those who felt that any civilization capable of such high-technology specialization must inevitably self-destruct, it gave a faint glimmer of hope, if hope it was for a planet to become a segment of a computerized cosmic organism.

Another article discusses self-awareness, described as "program-awareness." It is dull, but Fuller reads it to keep his worries still. The author holds that when a creature becomes aware of its program, then choice becomes possible where choice was not a factor before. Fuller knows the argument since it is written by a former student, a very civilized fellow. The cell block is quiet.

Finally, he dares to read again the news of his own imprisonment:

> FULLER, HERBERT T. Indefinitely detained for alleged subversion to internal security of the western world. Author of the famous Computerized Outlaw Prevention Program (COPP) that has practically perfected prediction of anti-social, criminal events. Noted lecturer and political advisor, president for many years of Computer Applications for Social Problems Institute (CASPI). Eccentric, famed for his hobby of raising biocomputer organisms, actually combinations of bacteria and ssmpu's which could perform circus tricks on a microscopic level. . . . ▼


ARTIFICIAL INTELLIGENCE
green
"SOFTWARE"
tan
"GARBAGE IN"
GARBAGE OUT"
shirt color red-orange
sizes S (30-32)
to XL (40-42)
"DOES NOT COMPUTE"
lt. blue
:SYSTEM UP AND RUNNING:
yellow

$5 each

COMPUTER T-SHIRTS

name ____________________

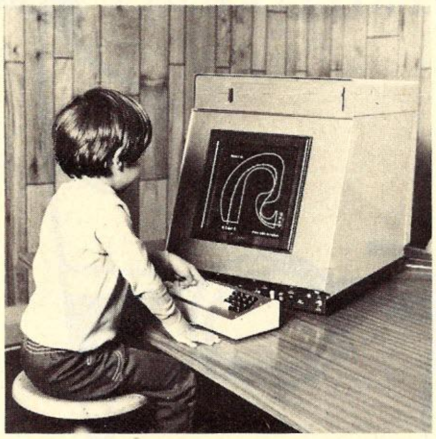
address ____________________

city ________ state ____ zip ______

mail to MARTHA HERMAN
114 W. 17 STREET
NEW YORK 10011
(212) 691-2821

quantity	chest size	design	$	
add .60 per shirt for shipping				
total enclosed				

PLATO Makes Learning



Mickey Mouse

by Elisabeth R. Lyman

PLATO is an interactive computer-based system developed in the Computer Education Research Laboratory (CERL) at the University of Illinois at Urbana-Champaign. The purpose of CERL research has been to produce a cost-effective computer-based educational system. We have aimed for power and flexibility in order to provide high quality interaction. PLATO therefore incorporates innovations in display technology, terminal design, communication hardware, system software, and courseware development. The result of seventeen years of research and field testing is, today, not only an advanced and sophisticated automated educational system, but also a national, indeed international, computer-based information and communications network.

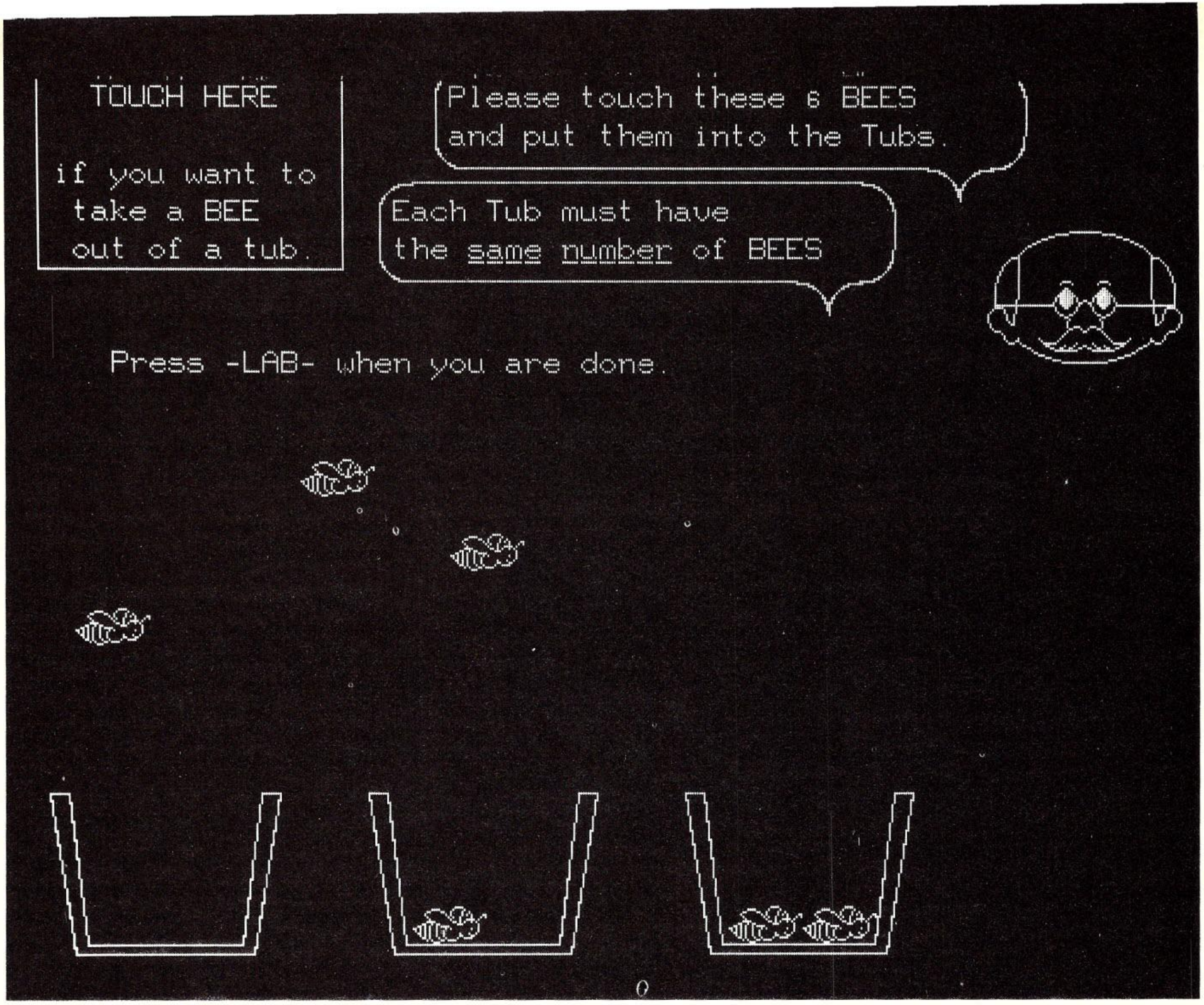
PLATO systems are presently installed at four locations: the University of Illinois, the Control Data Corporation at Arden Hills, Minnesota, the University of Quebec, in Quebec, Canada, and at Florida State University at Tallahassee, Florida. Additional systems are expected during the latter part of 1977. The systems operate independently, but can be linked together for intercommunication. Lessons, software programs, and communication information can be rapidly transferred from one system to another, offering an extensive exchange of teaching materials and information. Eventually it is hoped that all PLATO systems will form a grid of compatible regional networks.

The University of Illinois PLATO IV system supports approximately 950 terminals controlled by a central computer system in Urbana, Illinois. The 950 terminals are located at about 140 sites: twenty-six sites on the campus of the University of Illinois in Urbana-Champaign, nine elementary schools, two high schools, six community colleges, twenty government supported installations, twenty-seven medical sites (fifteen at colleges or universities), thirty colleges and universities, and about twenty miscellaneous business and industrial installations. The terminals are scattered from San Diego, California to Boston, Massachusetts and from Madison, Wisconsin to Wichita Falls, Texas. Over 500 of the 950 terminals have been active simultaneously.

A PLATO TERMINAL

A PLATO terminal is the interactive device which provides communication between a user and the computer. The terminal's major components are a keyset which transmits the user's request or input to the central computer, and a flexible visual display for presentation of lesson materials. The display is capable of graphics, symbols, alphanumerics, animation, and rapid selective display writing and erasing. Several peripheral devices may be added to a terminal, with standard devices including a random access slide selector, a touch device, random access audio, and a music

Photographs courtesy of CERL



TOUCH HERE

if you want to
take a BEE
out of a tub.

Please touch these 6 BEES
and put them into the Tubs.

Each Tub must have
the same number of BEES

Press -LAB- when you are done.

synthesizer. Others include remote printers, optical scanning, an electronic voice synthesizer, and research equipment developed or adapted to the PLATO system by PLATO users.

All University of Illinois PLATO terminals are equipped with plasma display panels which are flat panels also doubling as projector screens for images stored on microfiche. We have found that these are well-suited to lowcost mass production and enable economies in design and operation of the telecommunications and computer software of the system.

The plasma panel has a grid of 512 by 512 transparent conductors at whose intersections a neon discharge can be ignited or extinguished under computer control. The writing speed for text on such a display is 180 characters per second. Graphs and line drawings are displayed at a rate of sixty connected lines per second. The panel has inherent memory and does not require repetitive replotting. Selective writing or erasing of parts of the display is possible without disturbing the rest of it.

The principal PLATO input device is a keyset which has a standard set of typewriter keys plus special keys performing different functions in different lessons.

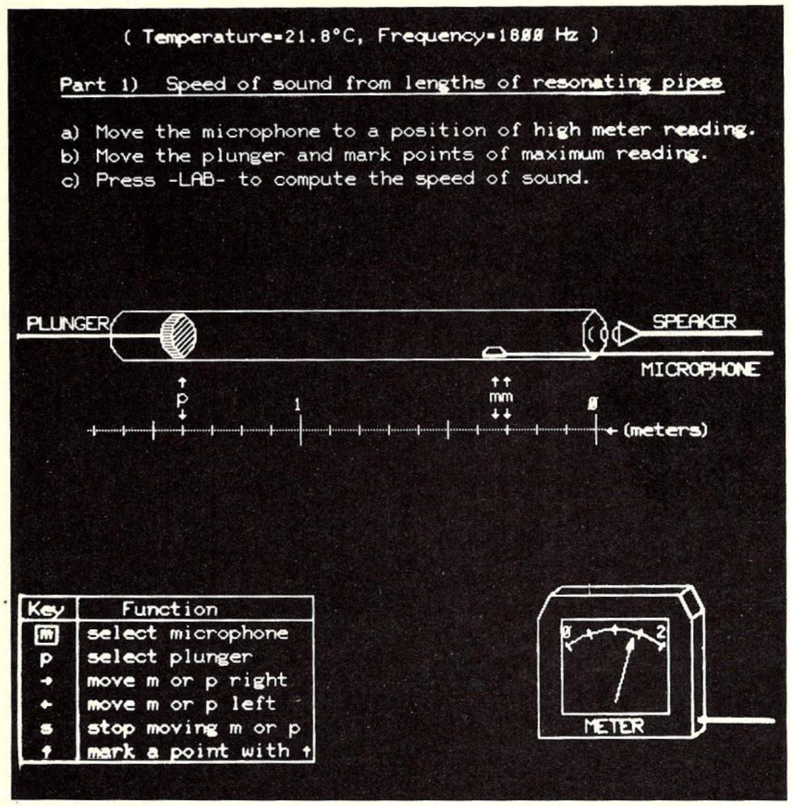
The terminal contains character and line generators and interface electronics to a 1200 bit-per-second telephone line. The character generator includes 126 standard alphanumerics and 126 characters alterable under author control. These characters can be loaded from the central computer with character patterns for special alphabets or pieces of pictures which are then to be presented on the plasma display panel.

A new terminal, the PLATO Programmable Terminal (PPT) is now in production. The terminal architecture is really a minature computer system which performs as a PLATO terminal because of the attachment of a plasma panel. The new terminal will be more economical than the existing PLATO IV terminal and will, hopefully, have greatly expanded capabilities and give improved performance.

PERIPHERAL EQUIPMENT

The random access slide selector uses the plasma panel as the screen for back projection of images of color photographs from a four-inch-by-four-inch microfiche with a 256 image capacity. The random access image selector is pneumatically driven and is under computer control. Since the slide images are rear-projected from inside the terminal, they do not interfere with the panel in any way and additional information can be presented to the student simultaneously on the panel.

The touch sensitive device defines a sixteen by sixteen grid of the display panel. When intersecting light beams are interrupted by a finger, the position of interruption is transmitted to the computer.



(Temperature=21.8°C, Frequency=1800 Hz)

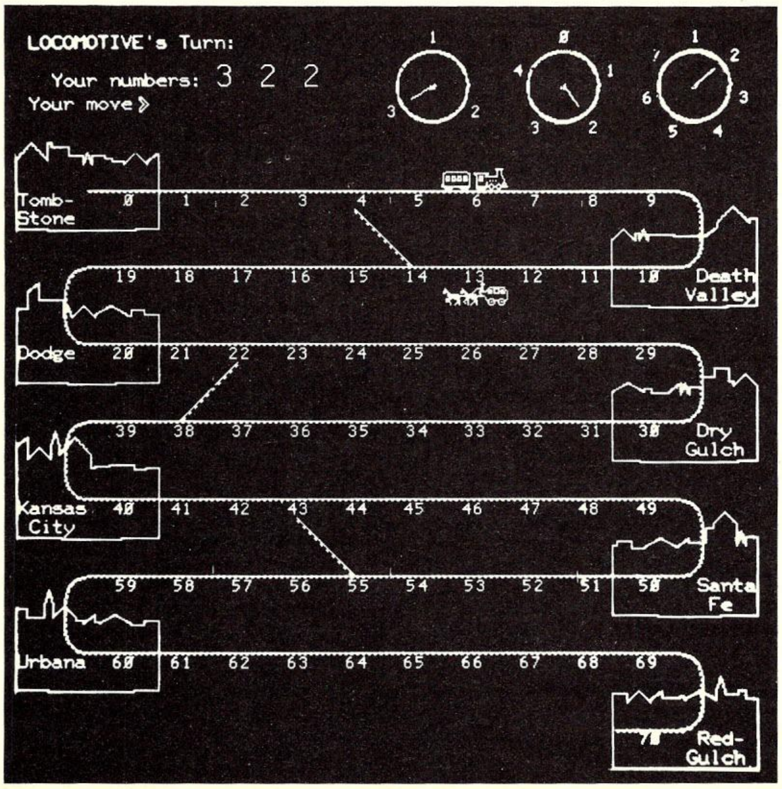
Part 1) Speed of sound from lengths of resonating pipes

a) Move the microphone to a position of high meter reading.
b) Move the plunger and mark points of maximum reading.
c) Press -LAB- to compute the speed of sound.

Key	Function
m	select microphone
p	select plunger
→	move m or p right
←	move m or p left
s	stop moving m or p
↑	mark a point with ↑

Sound waves in air can be measured and analyzed experimentally by computer as in this simulated physics lesson.

A locomotive and a stagecoach race in the computer game How the West Was Won. A wily computer can combine its chosen numbers in clever ways, so its human competitor had better develop a cunning winning strategy.



LOCOMOTIVE's Turn:
Your numbers: 3 2 2
Your move »

There are also ports available on the terminals for attaching external electronic devices such as audio or other special equipment. These devices may both receive data from the computer and send data to the computer.

The random access audio device is pneumatically driven. It allows instant playback of prerecorded messages stored on a fifteen inch interchangeable plastic disk. Over twenty minutes can be stored with an access time to any of over 4000 starting points of about one-half a second. Space can also be provided for student recording immediate playback and/or comparison to a prerecorded model.

The Gooch Synthetic Woodwind (GSW) is a four-voice music synthesizer which, when used with the appropriate PLATO courseware, reproduces music. The GSW can also be used for sound-generating experiments and as a sound source input to standard analog music synthesizers.

SYSTEM HARDWARE

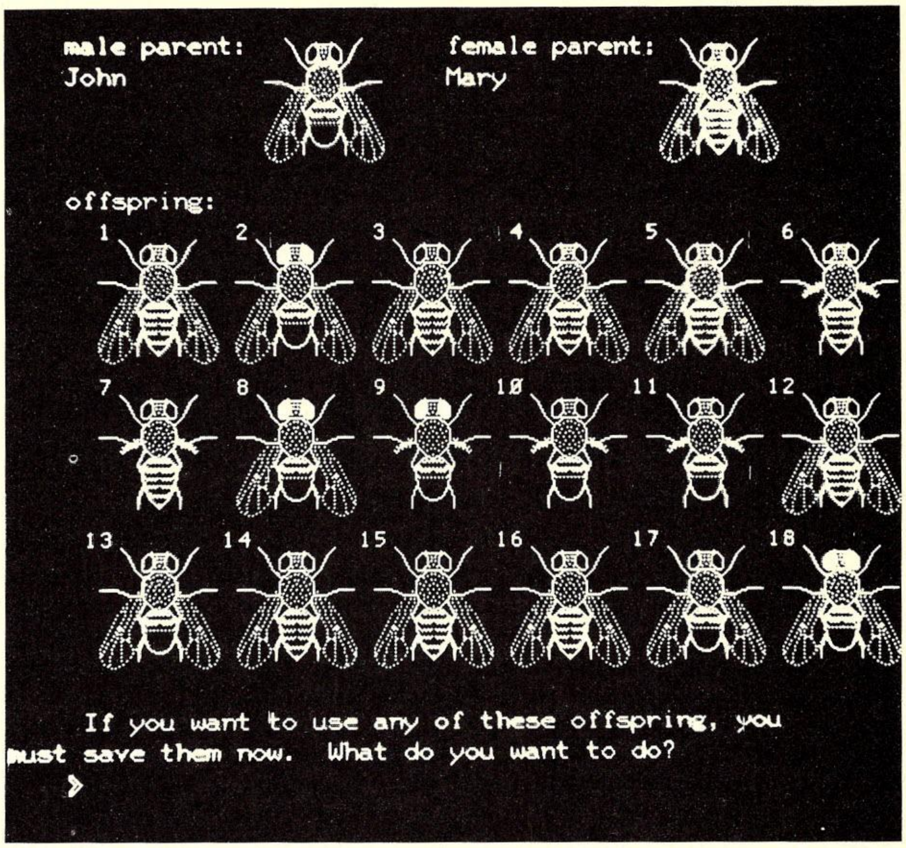
The heart of the PLATO computer system is not one, but two computers: a CDC Cyber 73-24 and a CDC6500. The permanent storage medium for the computer programs and lessons is a disk system from which are copied programs and data to and from an Extended Core Storage (ECS) of two million words. The programs and data in the ECS are then swapped in and out of the central memory for processing by the central processors with a transfer rate several hundred times faster than that of disks or drums and an access time a thousand times shorter than the waiting time to retrieve information from disks or drums directly to the central memory of a computer.

The access time of the ECS unit is actually less than five microseconds and the transfer rate is ten million sixty-bit words per second. ECS makes possible fractional-second response times (125 milliseconds) to hundreds of graphic terminals at relatively low cost. The electronic swapping memory (ECS) is much less expensive than a large central memory.

The communication between the computer system and the terminals is via an interface unit (CIU) onto a standard television channel or microwave link, and thence to site controllers each handling thirty-two terminals. The information from there is distributed to each terminal via a 1200, 4800, or 9600 bit-per-second telephone line. Terminal to computer communication is via a telephone line. A line concentrator at the site controller multiplexes the information from the thirty-two terminals onto the single telephone line and transmits it to the computer. Each terminal is guaranteed an average of 2.3 inputs per terminal which is five times more than the measured average.

System reliability is high. More than ninety-five percent of the total scheduled time is available to the users. The cost objective of the system, although not yet met, shows progress despite inflation.

Hardware developments now in the research stage include a low-cost replacement for core-based swapping memory, improved communication devices, portable terminals, and voice recognition and synthesis capabilities.



If you want to use any of these offspring, you must save them now. What do you want to do?

>

Fruit flies breed on video display in a simulated biology laboratory. The genetics student chooses the parents; the computer generates drawings of the resulting offspring, complete with the right genes.

EDUCATIONAL PHILOSOPHY

A primary purpose of a computer-based teaching system is to provide an automated means for individualizing student instruction while serving large numbers of students simultaneously. The teacher selects a curriculum from the available instructional material, and the computer presents this material to the students while monitoring and evaluating their performance. Each student can work at his own pace receiving instantaneous reinforcement for correct work and having access to special information and help when problems arise. The lesson material can be easily rearranged by the teacher or revised by its author in order to improve or modernize instruction. In addition, the teacher is freed for special work with students, an advantage which conventional teaching does not usually include.

The versatility of the system provides for presentation of drill and practice routines, tutorial material, problems to be solved, information to be retrieved, simulated experiments, dialogues, or computations. PLATO lessons aim to accommodate individual rates and styles of learning, as well as to provide acceptance of student-constructed answers, immediate feedback for student responses, and remedial or advanced instructional material. The instructional material is written and edited on-line from any terminal while timesharing the system with other authors and students.

A programming language, TUTOR, was created especially to minimize the need for specialized knowledge of computer programming. It includes a variety of graphic and computational features, a natural language answer-judging capability, and many branching commands which give the teacher computer control of teaching style and strategy. Production and revision of PLATO lessons are also facilitated by a range of editing aids and other system authoring features.

Instructors may either write their own lessons or assemble a curriculum for their students from a catalog of already available lessons. Students may be specifically routed through a series of lessons or allowed freedom of choice from a catalog.

Any PLATO user can talk on-line to any other user anywhere in the world.

APPLICATIONS

PLATO terminals have been used in public and private schools, community colleges, public and private universities and colleges, military training centers, correctional institutions, and industrial sites teaching a variety of subjects. Three million contact hours of use on the system were logged between July 1, 1974 and April 1, 1977 alone.

No age has been found too young or too old for using the system. From grade school children who learn reading and mathematics to adults continuing postponed education, from children with learning disabilities to inmates of correctional institutions, from persons from other lands trying to learn the English language to graduate students in law or physics—all these and more have participated in computer-based education.

Users have access to about five thousand hours of instructional material in over 140 subject areas. There are programs in engineering, pure sciences, foreign languages, social sciences, all levels of mathematics, natural sciences, business, medical and health sciences, english, and psychology to name only part of the list. Simulation studies of social, biological, or technological systems abound. Game playing, both instructional and purely recreational, helps increase the popularity of the system.

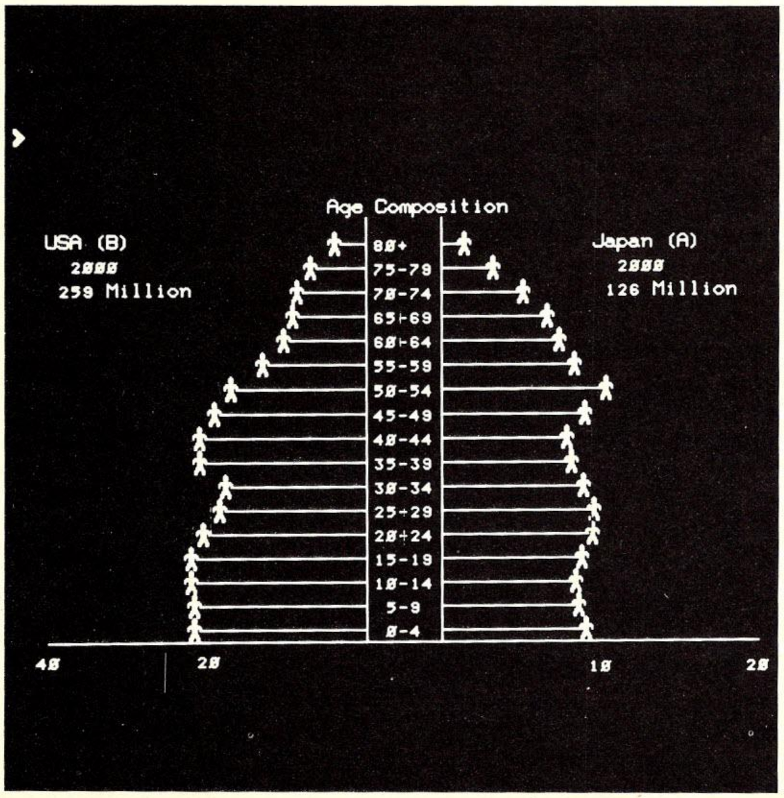
Instant information exchange is one of the most important features of a system which started out as only an aid to teaching.

The highly interactive character of the PLATO system allows all kinds of interterminal communication. Any user can "talk" on-line to any other user no matter where he is located. Messages to a user at another terminal appear on the user's plasma panel after he has responded to being paged. The two users then communicate by typing messages which both users can see. A user can monitor another user's screen on request. Such a capability is especially useful for authors taking advantage of the on-line consulting service available to them.

Numerous notefiles exist such as private ones for personal communication, group notefiles which serve as forums for users with special interests, and a general notefile in which authors can ask questions about or make suggestions for system changes. Significant improvements in the TUTOR language have been the result of user suggestions.

Easy communication allows students to contact teachers, make comments on lessons, and receive special messages relative to their instruction. Consultants can aid authors who have programming problems. Messages can be transmitted to everyone on the system or to one individual. Rapid, even instant, information exchange is one of the most important features of a system which started out exclusively as an aid to teaching.

In typical demographic graphics, a PLATO computer projects the age composition of the USA and Japan to the year 2000.



A number of commercial ventures have also resulted from various facets of the University of Illinois PLATO system. The University controls the patent rights to the PLATO system and its components. Non-exclusive licensing to commercial firms is now returning a part of the taxpayer's investment.

Several large companies are directly involved with PLATO. Magnavox produces PLATO terminals and has supplied the PLATO IV terminals for the University of Illinois system. Control Data Corporation, a long-time supporter of PLATO research, is licensed to sell PLATO systems and PLATO service, and to manufacture PLATO terminals.

A variety of applications have been found for plasma panels, ranging from small narrow readout displays to multiple screens on a terminal which can simulate several types of graphic terminals. The displays are used in banking terminals, key-to-disk systems, word-processing systems, and in small devices such as cash registers. Plasma display devices are manufactured not only in the United States, but in Japan.

Many private businesses have already found applications for the PLATO system. Among these users are a real estate management firm, a hospital equipment repair service, a large retail and mail order department store firm and a company which manufactures heavy-earth moving machinery. The real estate firm keeps all its rental and lease records on the system. The repair service maintains an up to the minute inventory on the location and availability of its specialized replacement parts. Employee training courses for specific positions have been developed by the retail store whereas the machinery company provides educational benefits for its employees by offering them instruction via the PLATO system.

SERVICES

The PLATO system presents a broad set of services to its users in addition to the instructional service which was initially its primary function. Applications beyond the instructional ones are diverse and include:

- Electronic note exchange (with text, graphics, and animation)
- On-line communication (with text, graphics, and animation)
- Entertainment (games, music, simulations, etc.)
- Personal services (medical, financial, psychological, educational, and career planning)
- Research computation
- On-line research (analysis of physical experiments controlled by a terminal; also educational and social on-line, real time research)
- Data processing
- Information retrieval
- Word processing

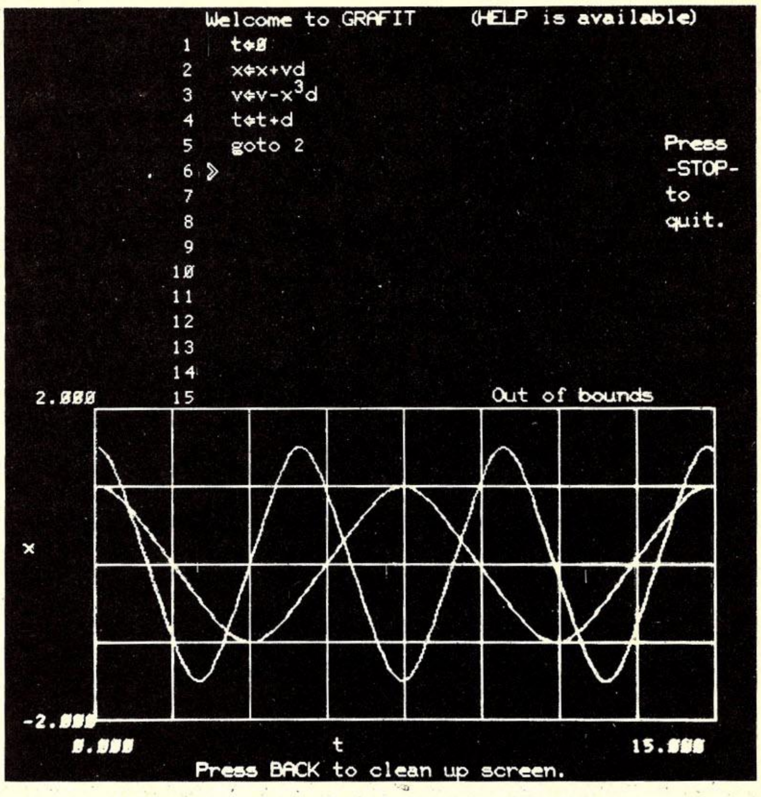
THE PLATO SERVICES ORGANIZATION

The PLATO Services Organization (PSO), a division of the Computer-based Education Research Laboratory, offers on-line consultation and maintains an extensive encyclopedia about all aspects of the system with emphasis on its programming language, TUTOR. It also offers curriculum information and training to individuals and groups of PLATO users. The training includes:

- Orientation to the terminal and keyset
- PLATO communications features
- Editing
- Introduction to the TUTOR language including unit structures of TUTOR, creating displays, calculations and variables, conditional operations, branching, and judging student responses
- How to use AIDS, the on-line informational encyclopedia and manual
- Other features of the system (determined by the interests of the participants)

EVALUATION

The main objective in the development of the PLATO system has, of course, been educational. Evaluation of the PLATO method of teaching is facilitated by the ease of storage of student response data as it is produced, and by a wide variety of lesson management tools available to instructors for interpretation of student progress and achievement. Despite the enormous number of variables involved in instruction, analysis of PLATO lesson effectiveness has been both subjective and objective.



```
Welcome to GRAFIT     (HELP is available)
 1   t⇐0
 2   x⇐x+vd
 3   v⇐v-x^3d
 4   t⇐t+d
 5   goto 2                                Press
 6 >                                       -STOP-
 7                                         to
 8                                         quit.
 9
10
11
12
13
14
15                         Out of bounds
Press BACK to clean up screen.
```

GRAFIT, a simple algebraic programming language, puts mathematical plotting on display. Both the programs and the parameters can be altered without disturbing a previous graph, thus allowing the charting of superimposed curves.

Teachers report that students do as well or better in PLATO-based courses and particularly in instances where homework is involved in the PLATO instruction. Students seem more motivated to complete assigned PLATO exercises. Authors who have written unique teaching presentations often point out that their PLATO lessons teach special concepts far better than by any previously used techniques. Non-author teachers and their students agree that PLATO materials are useful.

> ***Students do better in PLATO-based courses, particularly in instances where homework is involved.***

Various studies indicate that over ninety percent of students enjoy PLATO instruction and believe it is helpful in learning material. A large number have now used PLATO in more than one subject area over several years so their reactions no longer mirror solely the effect of the novelty of the system. Although there is a consensus that the system should not completely replace the human teacher, PLATO teaching has had a promising and enthusiastic reception on the part of teachers and students.

An independent evaluation of the PLATO teaching system was carried out in 1974 and 1975 by the Educational Testing Service, Princeton, New Jersey, for the National Science Foundation. A report of this evaluation is scheduled for completion soon. ▼

CHARGED COUPLES

by Sandra Faye Carroll

At this very moment, somewhere out in the rippling heat of the Texas desert, there is a geologist searching for oil. Perhaps he is taking soundings of the landscape, laying down a string of forty-eight sonar stations, and then recording the echoes from each station onto its own track of a forty-eight track audio tape. The sounds bounce off the internal anatomy of the desert, describing the water, empty holes, shale, rock or oil down below.

Through the dust and the heat, the truck bounces along, reviewing each station along the stringer. In the heat, the expensive precision tape recorder may break down or the tape itself may stretch or break. The truck is equipped with electricity-guzzling air conditioning, not to keep the technician dry and comfortable, but to keep the tape recorder cooled to an optimal working temperature.

Soon, one of these geological surveying companies will have data-processing trucks equipped with Fairchild Semiconductor charge-coupled digital memory devices instead of tape recorders. The charge-coupled device is very small, and has no moving parts, so vibrations will not throw it out of whack. It can withstand heat up to 70 degrees centigrade, which comes to about 167 degrees Fahrenheit. So goodbye air conditioning units. Whereas a tape recorder can store about one million bits of information, using quite a lot of electrical power in the process, one CCD can store a little more than 64 thousand bits. (64K, which really means 2^{16}, or two multiplied by itself sixteen times, comes to 65,536, very roughly rounded off to 64K—K for kilo from the Greek khilioi, a thousand.) Each truck will hold eight boards, and each board will hold one thousand CCD chips. So, if one CCD stores 64K bits, then a single board of one thousand CCDs stores 64 megabits, while using about the same amount of power as a 100-watt light bulb. Eight boards will give the Texas geologists about ten times the memory capacity they have had, more reliably, and at less than half the total cost.

Charge-coupling has nothing to do with joint credit cards or wife swapping. Coupling in electrical terms means, essentially, transferring energy from one place to the next. In charge-coupling, a bunch of electrons (not a stream of electrons as in a current) travels from one end of a string of bit locations to the other, like a submarine sliding below the surface of the silicon chip instead of running along the surface through some sort of conductor as in more conventional equipment. A charge-coupled device is made up of a series of strings, each holding within them a specific number of locations for bits. These strings, called shift registers, take up most of the space on a CCD. Fencing these in on each chip, however, are some standard metal oxide semiconductor instrumentation devices, to provide voltage feedback lines, small amplifiers, and level shifts to shrink current down to small enough amounts for the CCD to digest and then boost it up at the other end of the CCD so that the rest of the equipment can sense it.

Charge-coupling has nothing to do with joint credit cards or wife swapping.

The shift register is the core of the CCD. Each shift register contains a set number of wells to hold the electrons. These potential wells are not indentations in the silicon, the way a water well is an indentation in the earth, but they are, nonetheless, areas into which electrons fall or are drawn, much the way rain water falls into a sewer. The sewer lies gaping, always ready, whether or not there is rain.

But a potential well is transient. The electrode lying above it creates it anew every few microseconds, whenever a voltage is applied to the electrode. Since a potential well is intangible, it has no sides or bottom. The voltage in the area beneath the electrode attracts electrons to its vicinity and holds them only as long as its electrode is on. Turn off the electrode and the electrons wander off aimlessly in all directions, attaching themselves to various available silicon atoms as well as dropping off into various traps and defects lying in wait. A regularly flashing voltage, about a million pulses per second, acts as a clock that turns each electrode on and off in turn. The clock voltage comes from apparatus outside the CCD itself.

A potential well is intangible; it has no sides or bottom.

An input diode stands at the beginning of each shift register. Current, i.e. electrons, comes in through the input diode. Next to the input diode sits an input gate. The gate is open when voltage is applied to it, and closed when the voltage is gone. When the gate is closed, the current from the input diode circles endlessly from source to diode and back.

ROLL OUT THE BARREL

At first, the input diode is laden with electrons, the input gate is closed, and the first electrode on the other side of the input gate is on. The electrons in the input diode current are pulled a bit by the potential well created under Electrode Number One, but that potential well is further away from the electrons than the diode, so the diode's pull is stronger. When the input gate is turned on, the pull of the input gate, plus the pull of Potential Well Number One together are greater than the pull of the diode, and some electrons pass into Potential Well Number One.

(Imagine the electrons under the input diode to be beer filling a barrel set up on a table. The barrel is attached by a hose to a keg sitting nearby on the floor, and the beer wants to flow through the tube from the high barrel into the low keg. Gravity is tugging at it. But we've got the valve on the barrel closed so it can't go anywhere. When we open the petcock, the beer flows as fast as it can from the barrel to the keg, slowed only by its own viscosity and the friction inside the hose.)

Once Potential Well One is filled with electrons, the clock closes the input gate, and we have a discrete packet of electrons, a charge, trapped in Potential Well One. We can't leave it sitting there, however, or it will leak away. So we, or rather the clock voltage, immediately (a millionth of a second later) turns on Electrode Two, and then shuts down Electrode One.

ROMtutorial ROMtutorial

Bit: The simplest unit of information. It can be on or off.

String: A data structure which groups a number of characters into a sequence.

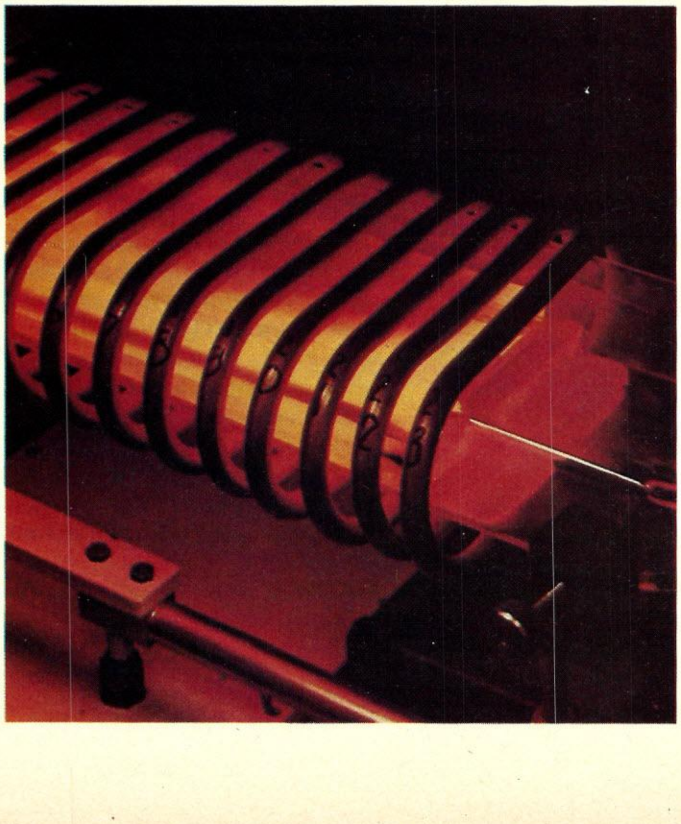
Chip: Integrated circuits are literally cooked into wafers of silicon three or four inches in diameter. Hundreds of IC circuits are repeated on a wafer. Later, each wafer is broken up into chips each containing one integrated circuit. Often the IC is called a chip after the actual chip is embedded in a plastic body.

Electrode: A metallic contact used to make electrical connection to something. A very general term.

Diode: Usually a semiconductor device widely used in electronic circuitry. It has the property of only allowing an electrical current to flow through it in one direction.

MAKING MEMORY

Photo series by Fairchild Camera and Instrument



In the making of integrated circuits such as those for CCDs, an oxide layer is grown on the surface of each silicon wafer. This is achieved by subjecting the silicon to an oxidizing atmosphere at extremely high temperatures in a reactor.

ROMtutorial ROMtutorial

MOS devices: Metal-Oxide-Semiconductor devices. Refers to electronic devices fabricated using a semiconductor technology based on depositing thin layers of metallic oxides on semiconductor materials. MOS technology enables devices to be made which consume very little power and can be densely packed on an integrated circuit chip. Almost all microcomputer technology is based on MOS techniques.

Random-access memory: Memory like a set of pigeon-holes, into any of which the computer can put new information or from any of which it can read old information. The computer can choose any pigeon-hole (or address) at any time.

The electrons from One are quickly drawn into Potential Well Two, the way iron filings will swoop from one electromagnet to another if you turn on the second magnet and then shut off the first.

As the electrons move from one potential well to the next, in bucket brigade formation, some few electrons stay behind in each well, as if a bucket retained a few drops of water each time we poured its contents into our neighbor's bucket. At the end of the shift register, an MOS amplifier, made of transistors and other conventional devices, boosts the current back up to its beginning level. The current returns to the input diode via non-CCD feedbacks to begin its quick trip through the shift register all over again. This is one bucket brigade which mustn't stop until the need for it is past, because the buckets have small leaks and, left to stand quietly full, their contents will seep out. However, if we pour the buckets from one to the next fast enough, we won't lose an appreciable amount.

We could store just one bit of information in each shift register, but that's mighty inefficient, all of those potential wells standing empty until the bit comes for a short visit. Why not fill them all and shift the whole string of bits continuously down the line? CCD is a digital memory, which means that a bit is either a 1 or a 0, either a packet of electrons or a hole. In our example, we fed in a 1, a packet of charge. Let's say we wanted to feed in 10011100. . . . While we move our packet, our 1, to Potential Well Two, we shut the input diode down for a millionth of a second, open the input gate and Electrode One, and shift some nothing, a hole, into Potential Well One. Then, the next millionth of a second, the clock voltage turns on Electrode Three, creating Potential Well Three, a millionth of a second later turns off Electrode Two and draws our packet (our 1) into Potential Well Three. The clock again shuts off the input diode, turns on the gate and Electrode One, making a hole in Potential Well One, and leaving us with a zero in Potential Well One, zero in Two, and 1 in Three. And so it goes, on through the digits, turning on and off the input diode in response to 1's and 0's, turning on and off the electrodes to create potential wells and obliterate them.

When the string of bits reaches the output diode, it reads them, applying a criterion based on two thresholds. If the packet of electrons is below the lower threshold, say one-quarter full, the output diode reads it as a zero and sends it to an MOS amplifier to be

At the start of each wafer-processing step, the silicon wafers are coated with photosensitive resin, which is applied in a spinner to assure even coating. The photoresist-coated wafers are then exposed to a mask containing part of the pattern of the desired circuit elements. These precision masks must be very accurately aligned by operators using microscopes. Exposure to ultraviolet light, which hardens the exposed photoresist, is performed in the same alignment machine.

drained until it is really empty. If the packet is above the top threshold, say three-quarters full, the diode reads it as a 1 and sends it to the amplifier to be boosted back up to full. Anything that falls between the two thresholds shouldn't. If the information must be changed in the string, it is altered in its proper order as it goes through the MOS feedback between the output and the input diodes.

SLOW TO REMEMBER

In order to get a piece of information out of this memory, the CCD has to run through the entire shift register looking for that bit in its place in line. For, you see, the CCD memory is serial. There is no way to pluck out a bit from the middle, without touching any other bit in the memory, as there would be in random access memories. This makes the CCD memory rather slow. If we put one thousand bits of information into a CCD, and wanted to take them out in a different order, say 1,3,2,4,6,5 . . . . the machine would spit out 1, skip 2 for the present, give us 3, then 4, skip 5, etc. and run through the whole thousand bits, coming back to 1, and then finally giving up bit number 2.

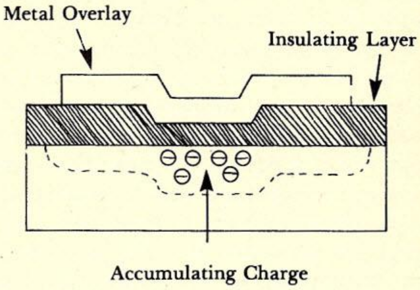
Even the CCD's memory density, its ability to hold at least four times as many bits of information as a RAM in the same size chip, works against its speed, for there is a venerable and vulnerably inviolable rule that says that the denser a memory is, the longer it takes to surrender its information. A CCD takes almost one hundred times as long to part with a particular bit as the high-speed random access memory does. But then, not all memory has to be fast in a computer. There is a lot to store in the machine while it is running, and there is room for memories of different speeds. High speed memory, the RAM type, is very expensive, and so one doesn't have very much of it. If we were running some material from a slow source like a punch card reader handling only two or three cards a second, it wouldn't make sense to use RAM to receive this information. We would run the material into the inexpensive CCDs, and when all of it had been entered into them, we would then empty the CCD memory into the main (RAM) memory, at a slower rate, perhaps, than the RAM could run, but fast enough nevertheless. Or we would just run the information off the CCDs because we wouldn't need permanent computer storage—we would already have the data on the punch cards.

Although CCD memory is slow for accessing—getting at a particular bit

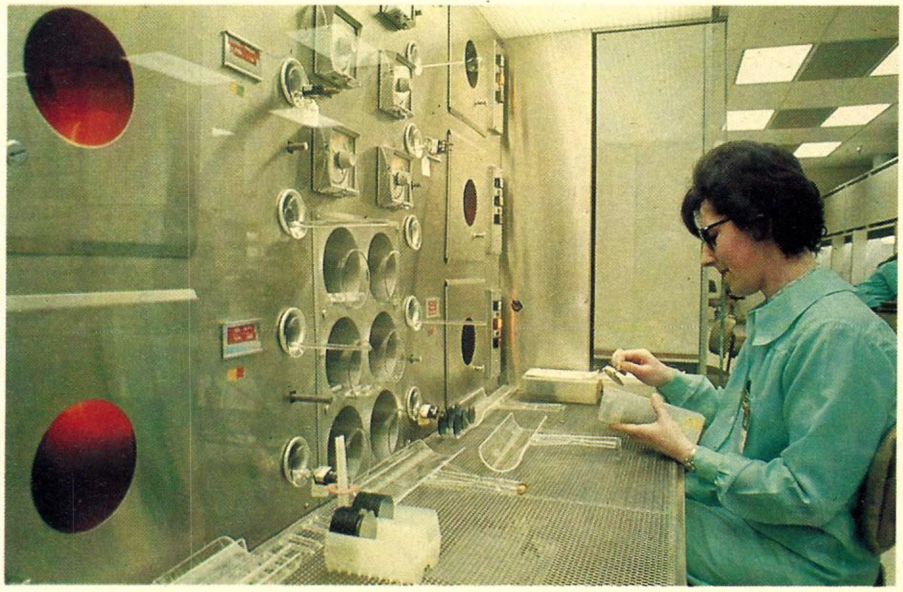
ROMtutorial ROMtutorial

Cursors: Pointers or windows used in graphic display devices to single out information from an image being displayed. Cursors can be moved by pressing keys or manipulating a joystick.

Array: A matrix or table of numbers. An array may also be a list of numbers.



In a CCD the semiconductor layer is separated from the metal overlay by a thin insulating layer. Voltage applied to the overlay causes a charge to accumulate in the "potential well" below. This charge can then be moved from cell to cell, forming memory patterns.



After masking, the unexposed photoresist is washed away, and wafers are etched to open holes in the surface oxide layer. The wafers are then placed in a diffusion furnace to allow specified dopants to enter the silicon and selectively change its electrical properties as necessary for a particular circuit component.

at a particular address in the memory—it is plenty fast enough when it is dumping a whole program into a main memory. That makes it a suitable partner in a large computer's time-sharing program. When several people work on a single computer at the same time, it appears that the computer is working with all of them at once. In fact, the computer can only do one thing at a time, like the rest of us, but it does it so fast that we can't sense when it switches back and forth among several users. Since the RAM is so expensive, the computer might not have enough main memory to store all of the sharers' programs at the same time. So, faster than the blink of an eye, it empties its RAM into some temporary storage, like CCDs, puts the first person's program on its RAM, works on it for a microwhile, dumps this program onto a CCD memory, and puts the second person's program onto its main memory to operate on it for a few fractions of a second, continuing to swap back and forth among programs while taking advantage of the density of the CCD memory—at least 16K on a chip the same size as a 4K RAM.

CCDs could also be used for the memory in smart terminals. On these terminals, which type something, move their cursors around the array, insert words, rearrange lines and move them about in the text, some sort of temporary memory holds the text while it is being worked on. When the text is completed and final, the operator pushes a button on the terminal and sends the text to the computer's main memory for permanent storage. The CCD memory is plenty fast for the smart terminal and its storage functions are merely temporary.

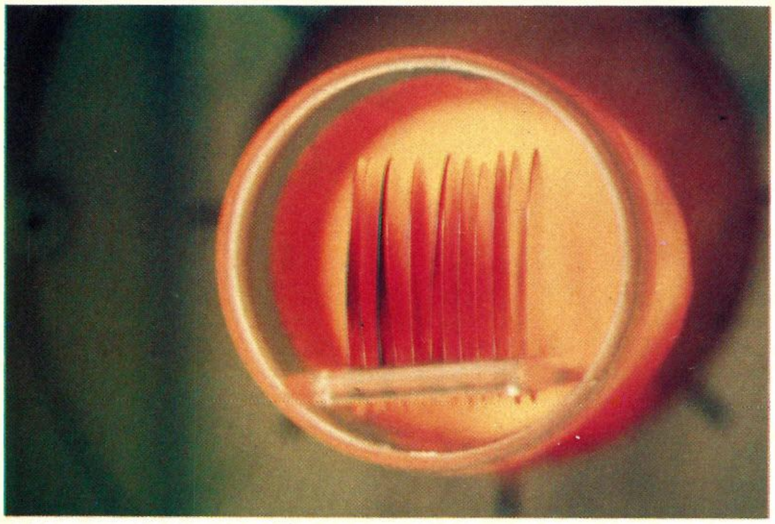
CCD is the quintessential volatile memory.

THE DISAPPEARING ACT

All CCD memory is by its nature evanescent. The clock keeps pulsing voltage to the electrodes, each shift register fills up and circulates its material endlessly, occasionally offering it up for necessary alterations or editing. The charge must be kept moving through the CCD or it leaks away. When the electricity is finally shut off, the electron packets, and therefore the program, evaporates. The quintessential volatile memory.

The inexpensive CCD memory costs only about a fifth as much as the conventional metal oxide semiconductor memories, and about a tenth as much as the high speed bipolar memories. Add to that the savings in power it uses, at one hundred milliwatts, about one fifth the wattage of an MOS and about one tenth the wattage of a B/P. (Selling price and power dissipation traditionally follow the same ratios.) Users who want density and low price for temporary information storage, and don't need speed, expect to use CCDs in drum and disk manufacture and to replace tape in products with slow retentive memories—the phone company's "the number you have dialed is not in service at this time," for example.

Even the manufacture of CCDs costs less than other wafer technologies. Bipolar, MOS, and CCD are all made by batch processing, casting perhaps 300 chips on a wafer or slice of silicon three inches in diameter and so thin that it floats on water. (Fairchild, as well as other members of the semiconductor community, plans to move to four inch wafers soon.) Although these slices are as smooth and as close to perfection as the crystal growers' art can conjure them, they nonetheless often have pinhole defects. In the diffusion processes of B/P and MOS technologies, any transistor sitting on a pinhole will be defective. Since each transistor is connected to the others, the failure of one means the failure of all, and that chip



Close-up of diffusion tube, showing wafers during a diffusion operation.

is useless. A large percentage of B/P and MOS chips are rejected by the manufacturer during the post-fabrication testing at the plant.

CHEAP CHEAP CHEAP

Only the border of a CCD chip contains MOS devices, (and only the MOS devices, by the way, use up power, because the power in the CCD portion is continually recirculated). Since at least sixty-five percent of the chip is CCD, and soon the proportion will be even greater, the chance that the pinhole will fall in the MOS area, under a transistor, is much smaller than the chance that it will fall in the larger CCD area where the charge passes under the surface of the silicon, oblivious to any tiny surface irregularities. With higher yield therefore, CCD production costs are proportionately lower.

Fairchild's CCDs do more than just remember. They see; they report. They monitor cable winding in a factory and count white blood cells under a microscope. They show the horizon to a pilot in an airplane, and advise on the uniform thinness of nylon thread. They read checks in a bank and feed the data onto video tape to save bank personnel the task of photocopying each check. And they sort dollar bills.

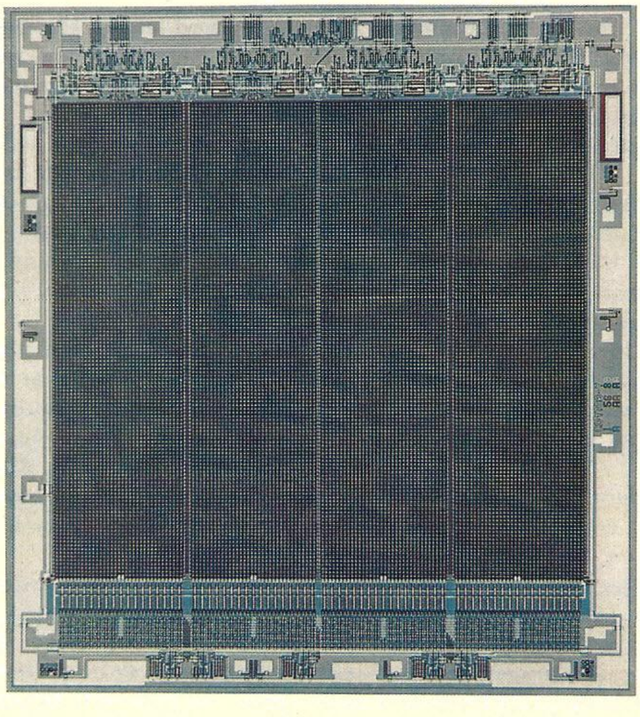
Because they are extremely sensitive to light, they can become the heart of video cameras able to take pictures comfortably in anything from bright moonlight to bright sunlight, a range much wider than that of conventional television cameras. Fairchild Semiconductor, in fact, has a black and white television camera a little smaller than a pocket Instamatic camera, which they can send down a small tunnel drilled into a collapsed mine shaft. The camera has a light on it and a lens which swivels, and through this tiny passage into the dark mine it can search for survivors and guide rescuers to them.

In the future, micro bucket brigades will be hauling around electrons in some very unusual places.

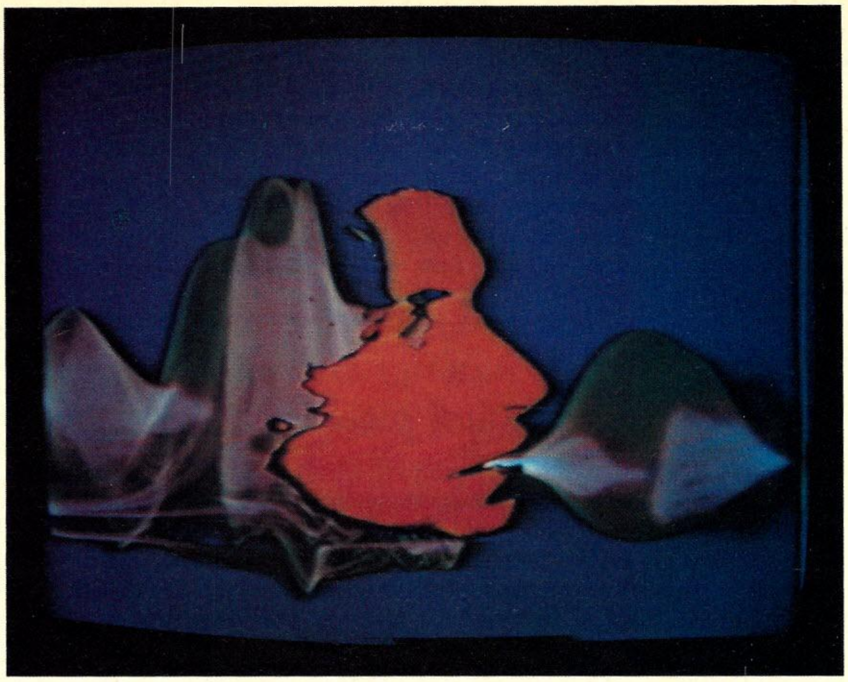
An imaging CCD operates much the same as a digital memory CCD, but the electrons come, indirectly, from light rather than from a current. A unit of light, a photon, enters the CCD through a little window covering the chip. The energy from the photon moves an electron from one of the inner bands of the silicon atom where it has been resting safe and secure to an outer band where the electron is kicked free in order to travel, with other liberated electrons, in packets just as if they had been introduced into the chip by an electrical current from a diode, moving along the shift register from well to well. The amount of electrons freed in a particular place on the chip is proportional to the amount of light that hits that locale, so the electrons fill the wells to a level that corresponds to the amount of light in their neighborhood. At the other end, they are finally detected, this time in gradations of blacks, greys, and whites, rather than in absolutes of 1's and 0's, and converted into an electrical signal that represents the optical image that originally hit the CCD. That signal, after going through some non-CCD translation, appears on a television screen just like any other form of video.

Memory and computer vision based on CCDs are here now. What they will bring in the future, beside scores of highly charged puns coupled to some rather abstract thinking, well.... At the very least, micro bucket brigades will be hauling around electrons in some very unusual places. ▼

The final step in wafer processing is metalization. To assure even coating, the wafers are placed in holders to be rotated in the metal desposition chamber. Masking and etching is then employed to remove unwanted metal and leave the correct interconnections in place on the finished wafer.



A completed 16,000-bit CCD.



Xeroxes & Other Hard

Bill Etra's graphics are real time generations with displays on a Cromemco Dazzler, Tektronix 4051, and Rutt/Etra Video Synthesizer. Mixing is through an Electronics Associates of Berkeley VideoLab controlled by an Altair 8080a communicating with a remotely accessed DEC PDP 11/03.

How do you take photographs off CRTs? The basic problem with photographing off a TV display, whether it's color or black and white, involves the fact that there are large sections of time when the picture isn't there. Basically, a TV scans half the picture in one-sixtieth of a second, displays the next half of the picture, and scans it in the next sixtieth of a second.

It blanks every time it finishes writing a line across the horizontal plane and when it's done writing half the picture lines or field, blanks itself again and starts over. Therefore, anything less than a one-thirtieth-second exposure will not give you a full frame, which means all exposures must be at least one-thirtieth of a second.

Since your camera is not synched to the TV set, it's a better idea, on a constant picture, to increase the exposure even further, to something around one-tenth of a second. This will eliminate the problem of photographing during a blanking interval—the results of which would produce a dark bar across your picture.

One-tenth-second exposures, however, cannot be taken without using a tripod. So your basic equipment must include a tripod as well as the camera itself.

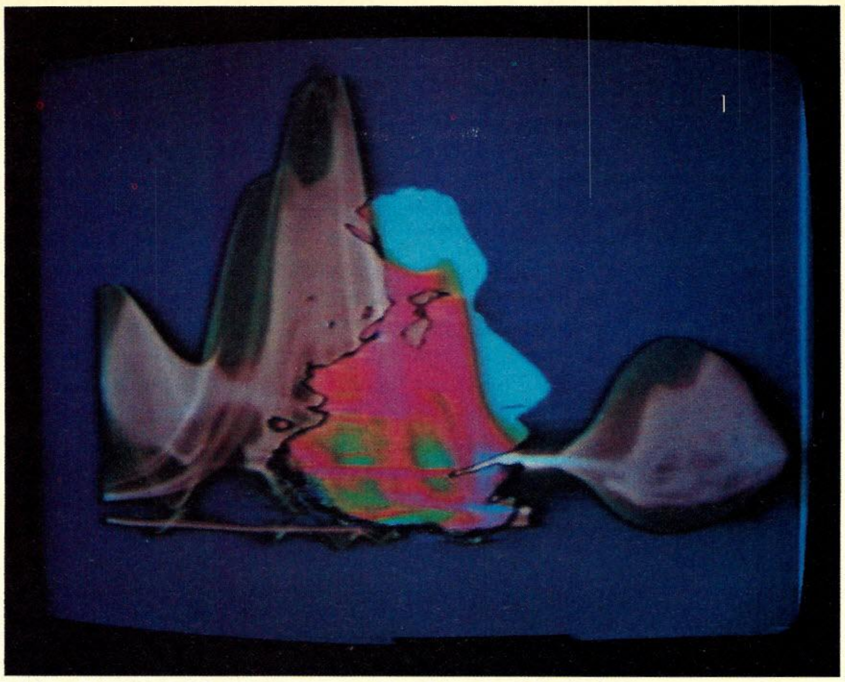
A word about the camera: Most people use a single-lens reflex camera. There are, however, two things wrong with single-lens reflex cameras for CRT copy work. One is that when the mirror flies up to take the picture, it bounces the camera, giving you a certain amount of shake, even on a tripod. The other thing is that they usually have a focal-plane shutter. This form of shutter is a slit which moves across the film plane. At high speeds it won't work at all, since the TV is writing in a similar plane and sometimes the two planes will be going at opposite directions. Although this is not a problem at very slow speeds it is still preferable to have a camera that has a leaf shutter.

If you are working with a standard single-lens reflex camera, it is a good idea once you have focused and tightened everything down on the tripod, to lock the mirror up. Most single-lens reflex cameras have a button that allows you to do this in order to minimize camera shake when you trip the shutter.

A second way of reducing camera shake is to use a cable release. A cable release is one of the least expensive pieces of camera equipment you can buy, usually priced under three dollars. Yet for this type of photography, it is also one of the most useful. A flash attachment, on the other hand, is totally useless. Flash pictures do not work off a CRT for obvious reasons, the CRT itself being a light transmission source.

When it comes to the actual shooting, the lens should be stopped down to compensate for the curvature of the CRT screen. Unless you stop down at least one f stop to increase the depth of field, the edges of the photograph will not be sharp. For instance, if you have, say, a 3.5 lens, you want your exposure to be at least 5.6. (As a side note, depth of field extends one-third forward and two-thirds back, so if you focus on the exact middle of the screen at wide open, and then stop down one, you will have covered yourself well for the curvature of the screen.)

CRT art programmed, executed, photographed, and Xeroxed by Bill Etra



Copy Off Your CRT

by Bill Etra

Another point to consider is that you don't ever want to turn the CRT up to full brightness or contrast. Most CRTs, being slightly out of focus, tend to bloom or fuzz the picture when in full brightness and full exposure. What you do want is a good medium range in both your colors and your exposure.

Speaking of color, the color temperature of a CRT is slightly to the blue of daylight, which means all pictures will tend to be slightly on the blue side when using daylight film. On the other hand, daylight film gives you the best results when photographing off the CRT. You can correct for the slight blue leaning by using the CRT's tint control to make the actual picture on the screen a little redder than you want in the final results. Color filtering is not really necessary.

For my own purposes, I shoot high speed Ektachrome daylight, a slide film I've found over the years gives me the best results. On a normally adjusted CRT, shooting with high speed Ektachrome film (ASA 160), the exposure at one-tenth of a second should be between f 5.6 and f 8. But that's just in the way of general guidance. Always use a light meter to give you exact measurements. When taking a light level reading from a transmission source, it is best to aim at the whitest, brightest part as opposed to an average part of the picture.

If you have a single-lens reflex camera with a split-level range finder, or ground glass type focusing in the center, you'll get a moire pattern when you are properly focused on the scanning lines of the TV image. The formation of the moire pattern is an easy way to determine that your focus is truly sharp. If you use another type of range finder, such as that on a twin-lens reflex, you need to focus on the actual lines.

As to the best type of television set for CRT photography, nothing beats a Sony Trinitron. It produces by far the highest quality color pictures, short of special monitors using red, blue, and green input directly.

Any of these methods can produce a high quality slide. When a good slide is enlarged by projection, you should be able to see the color lines of the TV

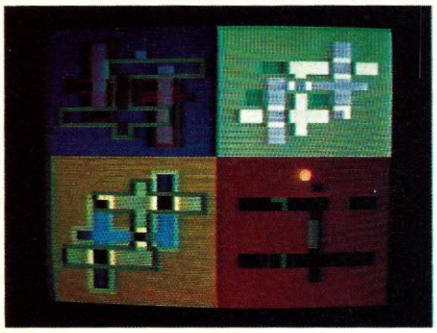
ROMtutorial ROMtutorial

Hard copy: Information in printed form, i.e. on paper. Soft copy would be information displayed on an impermanent medium, i.e. a TV screen.

CRT: Cathode ray tube. A television-style display device.

Color temperature: A method of describing the color of various shades of white, based on the temperature that a bar of iron must be raised to in order to produce the same shade. Color temperature is measured in degrees of Kelvin with Tungsten floodlights, for instance, having a color temperature of 3200 degrees Kelvin. Color temperature is extremely important in color photography where the sensitivity of the film to various colors is referenced to a particular color temperature.

Moire pattern: A pattern of (usually) curved lines, formed by the intersection of two superimposed patterns of regular lines. The shimmering pattern that results when two pieces of silk are superimposed is an example of a moire pattern. The effect is caused by an optical illusion which is not fully understood.



picture clearly. That is, after all, what you focus on.

What do you do with the slide once you've got it? Well, the easiest way to get a large print from your slide is to take it to your local color-Xerox shop. They can enlarge or reduce the image while copying the slide. The problem with this print-making method is that color-Xerox technology is not yet a perfected one. For instance, the operator may tell you that the slide does not look right, and that's the reason your copy is not the color of the original. This is not really true.

Color balance is the crucial fact, and you should never leave your work to be done while you're not there, especially if it's something as abstract as the graphics of a Dazzler or some other similar computer. With these, the Xerox operators have nothing familiar like flesh tones from which to judge the corrections of the reproduction and need you to specify the colors you want.

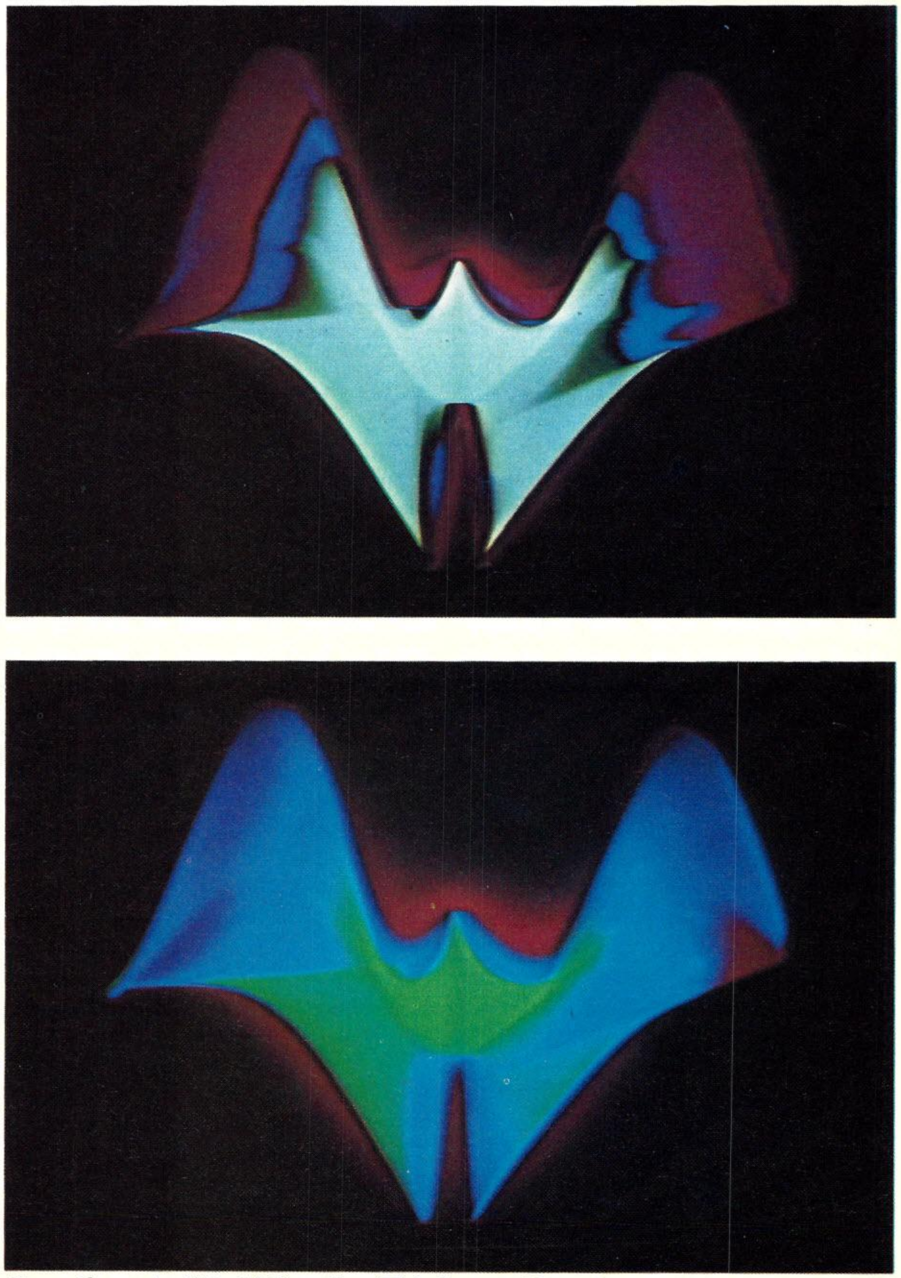
Incidentally, they do not pay for prints that fail. These are returned to Xerox for credit, so don't let them make *you* pay for them. If, as some few establishments do, they insist on your paying for low quality reproductions, take your business to another color Xeroxer. You have the right to refuse color Xeroxes if they don't come up to the original.

A nice touch of the color-Xerox process is that, up to a certain limit, you can use any weight or texture of paper. If you want a really fine linen reproduction, just bring the appropriate paper with you. It can easily be loaded into the machine.

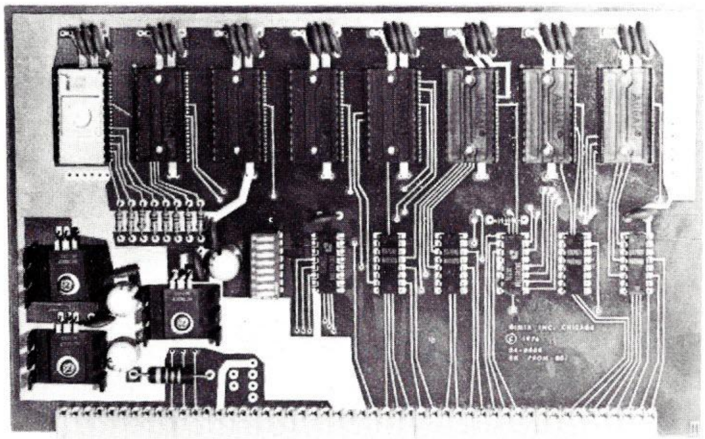
Another way of getting hard copy without going through so much hassle is to use one of several types of Polaroid cameras. Personally, I prefer the Polaroid SX-70 with the through-the-lens viewing for CRT photography. If you do use the SX-70 with through-the-lens viewing, note that the split range finder is *not* at center picture—it's lower than center picture. You must compensate for this or your picture will be cut off.

The SX-70 generally takes perfect exposures of a CRT in a darkened room. Its light meter will receive light from the surrounding area, and its electronic shutter seems to produce good exposures from the average TV picture. There are very few adjustments to make, other than in some cases putting your finger over the light meter intake, or putting various pieces of colored paper over it if the pictures are not properly exposed. Again, you should use a cable shutter release and a tripod.

With a regular Polaroid camera, the best choice is probably a 195 using pack film. It gives high quality color reproduction, as long as you remember that Polaroid film is a little more sensitive than others to the fact that a TV's color temperature does not precisely match daylight. Unless you remember to increase the tint almost all the way toward the red, it will tend to give you slightly blue pictures. Which, basically, is undesirable, unless perhaps you're working on snowy abstract art for your Christmas cards. ▼



From the series Would You Buy This Lot on Venus?

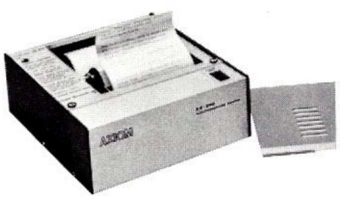


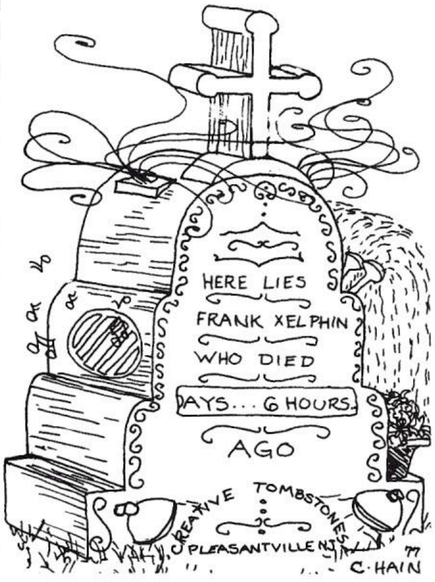
A HAUNTING THOUGHT

Thinking of computerizing your house? The SWTP compatable GIMIX GHOST boards are your answer. (GIMIX House Operating System Technology = GHOST.) Phone-conversation relay boards which convert private Touchtone phones into computer terminals, binary receiver boards that convert Touchtones into two tones, relay boards, and relay brackets will all keep things running smoothly while you're away and Casper's in charge. Complete with instructions and documentation from GIMIX Inc., 1337 W. 37th Place, Chicago, IL. 60609.

FOILED BY PRINTERS?

AXIOM's new 160 character/second line printer with a $655 price tag has a real sheen to it—it prints on aluminum-coated electrosensitive paper. Complete with infra-red low paper detector, bell, 96-character ASCII set and built-in self tester, the 80-column format printer even includes programmable character size. Comes in an RS232C serial interface model as well as at $740.





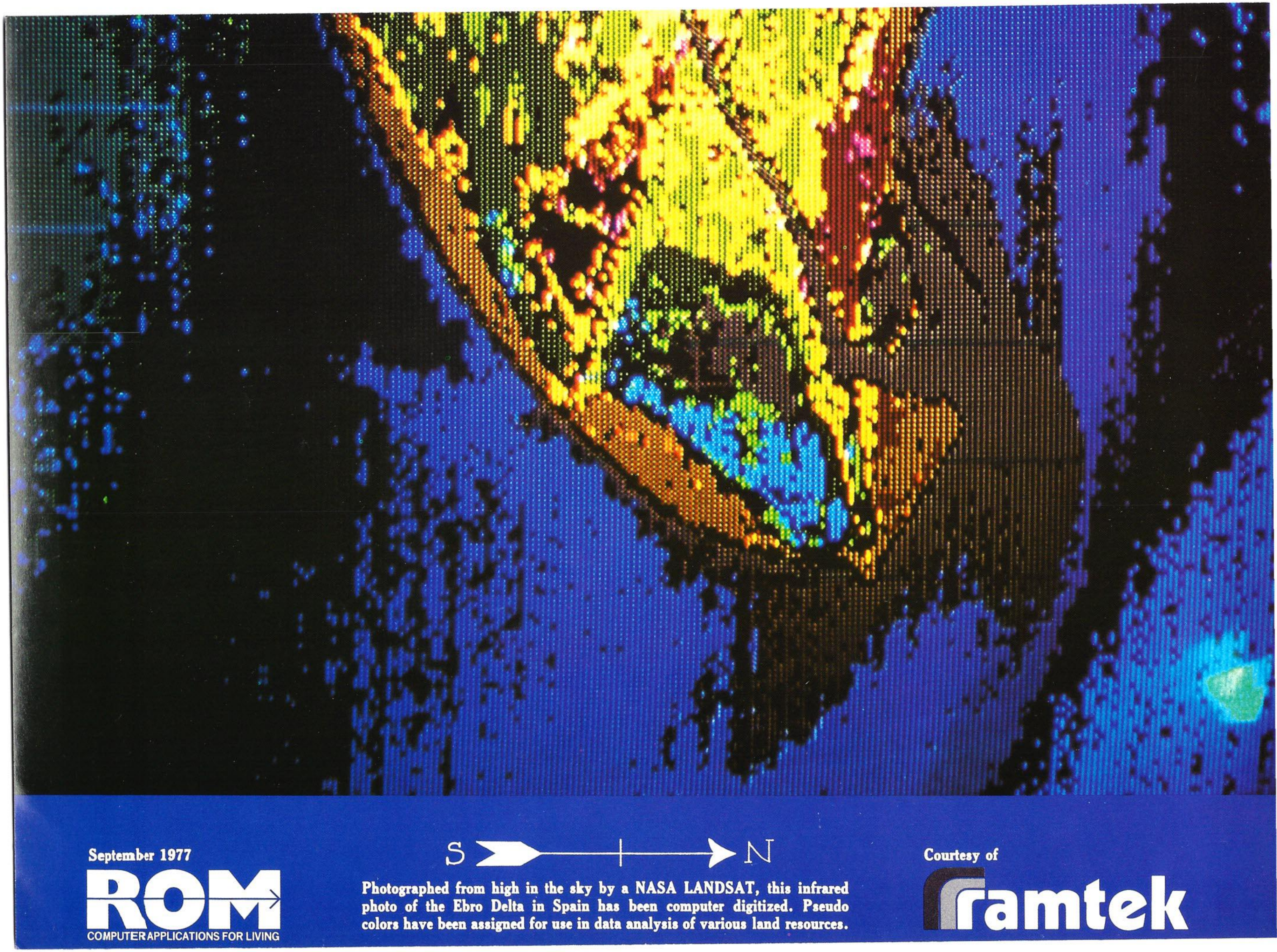
BOUNCY BURIALS

They've finally put it all together: solar energy, microprocessors—and granite. Featuring a proximity switch to insure a spectacular greeting from the beyond any time the bereaved approach the monument to their loved one, this "warm, personal, original" monument does everything short of whistling Dixie. Actually, that option is available, as are incense dispensers, synthesized music, and an automatic sprinkler system with rainwater cistern. There's even a digital readout panel for the calendar clock displaying, in sports-stadium style, the elapsed time since the passing of the deceased. "Lifetime guarantee," from Creative Tombstones, P.O. Box 66, Pleasantville, N.J. 08232.

HEATH HANGS SIXTEEN

The sixteen-bit personal computers are here, the parade starting with Heath Company's H11, available in kit form for $1295. The mastermind behind the module is Digital Equipment Corporation's workhorse, the LSI-11, using the PDP-11 instruction set, with DEC system software as part of the package. The H11 may be an easier kit to assemble than most, what with the sixteen-bit CPU coming fully wired and tested. In any case, those wanting a preassembled sixteen have a little longer to wait.

HEATHKIT H11



September 1977
ROM
COMPUTER APPLICATIONS FOR LIVING

Photographed from high in the sky by a NASA LANDSAT, this infrared photo of the Ebro Delta in Spain has been computer digitized. Pseudo colors have been assigned for use in data analysis of various land resources.

Courtesy of ramtek

Sol Terminal Computer
Processor Technology

One Sol-20 equals three computers.

To do real work with any computer, big or small, it takes a complete system. That's one of the nice things about the Sol-20. It was built from the ground-up as the heart of three fixed price computer systems with all the peripheral gear and software included to get you up and on the air.

Sol System I costs just $1649 in kit form or $2129 fully burned in and tested. Here's what you get: a Sol-20 with the SOLOS personality module for stand alone computer power, an 8192 word memory, a 12" TV/video monitor, a cassette recorder with BASIC software tape and all necessary cables.

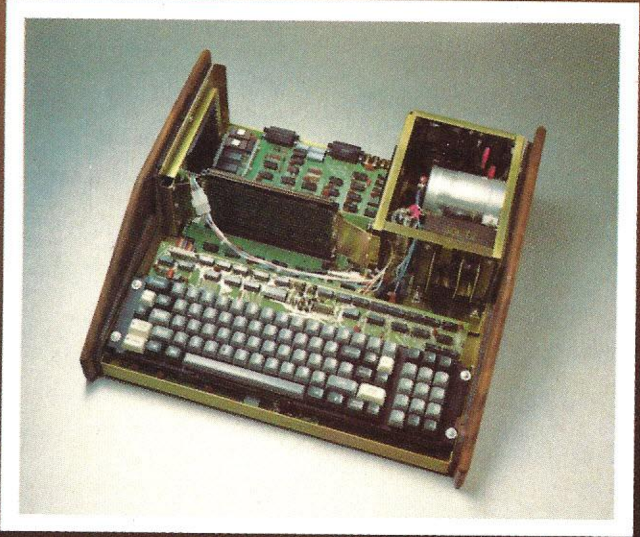
Sol System II has the same equipment plus a larger

Helios II
Processor Technology

capacity 16,384 word memory. It sells for $1883 in kit form; $2283 fully assembled.

For even more demanding tasks, Sol System III features Sol-20/SOLOS, a 32,768 word memory, the video monitor, Helios II Disk Memory System and DISK BASIC Diskette. Price, $4750 in kit form, $5450 fully assembled and tested.

And remember, though we call these small or personal computer systems, they have more power per dollar than anything ever offered. They provide performance comparable with mini-computer systems priced thousands of dollars more.



The functional beauty of Sol Computer Systems is more than skin deep. A look inside reveals a simple elegance of design and sturdy construction.

The Small Computer Catalog for the rest of the real computer system story.

Visit your local computer store for a copy of our fully illustrated 22 page catalog. Or you may write or call us if more convenient. Please address Processor Technology, Box O, 6200 Hollis Street, Emeryville, CA 94608. (415) 652-8080.

Processor Technology Corporation

See Sol here...

ALABAMA

ICP, Computerland
1550 Montgomery Hwy.
Birmingham, AL 35226
(205) 979-0707

ARIZONA

Byte Shop Tempe
813 N. Scottsdale Rd.
Tempe, AZ 85281
(602) 894-1129

Byte Shop Phoenix
12654 N. 28th Dr.
Phoenix, AZ 85029
(602) 942-7300

Byte Shop Tucson
2612 E. Broadway
Tucson, AZ 85716
(602) 327-4579

CALIFORNIA

The Byte Shop
1514 University Ave.
Berkeley, CA 94703
(415) 845-6366

Byte Shop of Burbank
1812 W. Burbank Blvd.
Burbank, CA 91506
(213) 843-3633

Byte Shop Computer Store
6041 Greenback Lane
Citrus Heights, CA 95610
(916) 961-2983

Computer Center
1913 Harbor Blvd.
Costa Mesa, CA 92627
(714) 646-0221

Data Consultants, Inc.
2350 W. Shaw, Suite 114
Fresno, CA 93711
(209) 431-6461

Bits 'N Bytes
679 S. State College Blvd.
Fullerton, CA 92631
(714) 879-8386

The Byte Shop
16508 Hawthorne Blvd.
Lawndale, CA 90260
(213) 371-2421

Opamp/Computer
1033 N. Sycamore Ave.
Los Angeles, CA 90038
(213) 934-3566

Digital Deli
80 W. El Camino Real
Mountain View, CA 94040
(415) 961-2828

The Computer Mart
624 West Katella #10
Orange, CA 92667
(714) 633-1222

Byte Shop
496 South Lake Ave.
Pasadena, CA 91101
(213) 684-3311

The Computer Store
of San Francisco
1093 Mission Street
San Francisco, CA 94103
(415) 431-0640

Byte Shop
321 Pacific Ave.
San Francisco, CA 94111
(415) 421-8686

The Computer Room
124H Blossom Hill Rd.
San Jose, CA 95123
(408) 226-8383

The Byte Shop
509 Francisco Blvd.
San Rafael, CA 94901
(415) 457-9311

The Byte Shop
3400 El Camino Real
Santa Clara, CA 95051
(408) 249-4221

Recreational Computer
Centers
1324 South Mary Ave.
Sunnyvale, CA 94087
(408) 735-7480

Byte Shop of Tarzana
18424 Ventura Blvd.
Tarzana, CA 91356
(213) 343-3919

Computer Components
5848 Sepulveda Blvd.
Van Nuys, CA 91411
(213) 786-7411

The Byte Shop
2989 North Main St.
Walnut Creek, CA 94596
(415) 933-6252

Byte Shop
14300 Beach Blvd.
Westminster, CA 92683
(714) 894-9131

COLORADO

Byte Shop
2040 30th St.
Boulder, CO 80301
(303) 449-6233

Byte Shop
3464 S. Acoma St.
Englewood, CO 80110
(303) 761-6232

FLORIDA

Sunny Computer Stores
University Shopping Center
1238A S. Dixie Hwy.
Coral Gables, FL 33146
(305) 661-6042

Delta Electronics
2000 U.S. Hwy. 441 East
Leesburg, FL 32748
(904) 357-4244

Byte Shop of Miami
7825 Bird Road
Miami, FL 33155
(305) 264-2983

Microcomputer
Systems Inc.
144 So. Dale Mabry Hwy.
Tampa, FL 33609
(813) 879-4301

GEORGIA

Atlanta Computer Mart
5091-B Buford Hwy.
Atlanta, GA 30340
(404) 455-0647

ILLINOIS

itty bitty machine co.
1316 Chicago Ave.
Evanston, IL 60201
(312) 328-6800

Reeves Communications
1550 W. Court St.
Kankakee, IL 60901
(815) 937-4516

itty bitty machine co.
42 West Roosevelt
Lombard, IL 60148
(312) 620-5808

INDIANA

The Data Domain
406 So. College Ave.
Bloomington, IN 47401
(812) 334-3607

The Byte Shop
5947 East 82nd St.
Indianapolis, IN 46250
(317) 842-2983

The Data Domain
7027 N. Michigan Rd.
Indianapolis, IN 46268
(317) 251-3139

KENTUCKY

The Data Domain
3028 Hunsinger Lane
Louisville, KY 40220
(502) 456-5242

MICHIGAN

The Computer Store
of Ann Arbor
310 East Washington
Ann Arbor, MI 48104
(313) 995-7616

Computer Mart
of Royal Oak
1800 W. 14 Mile Rd.
Royal Oak, MI 48073
(313) 576-0900

General Computer Store
2011 Livernois
Troy, MI 48084
(313) 362-0022

MINNESOTA

Computer Depot, Inc.
3515 W. 70th St.
Minneapolis, MN 55435
(612) 927-5601

NEW JERSEY

Hoboken Computer Works
No. 20 Hudson Place
Hoboken, NJ 07030
(201) 420-1644

The Computer Mart
of New Jersey
501 Route 27
Iselin, NJ 08830
(201) 283-0600

NEW YORK

The Computer Mart
of Long Island
2072 Front Street
East Meadow, L.I. NY 11554
(516) 794-0510

The Computer Shoppe
444 Middle Country Rd.
Middle Island, NY 11953
(516) 732-4446

The Computer Mart
of New York
118 Madison Ave.
New York, NY 10001
(212) 686-7923

The Computer Corner
200 Hamilton Ave.
White Plains, NY 10601
(914) 949-3282

OHIO

Cybershop
1451 S. Hamilton Rd.
Columbus, OH 43227
(614) 239-8081

Computer Mart of Dayton
2665 S. Dixie Ave.
Dayton, OH 45409
(513) 296-1248

OREGON

Byte Shop Computer Store
3482 SW Cedar Hills Blvd.
Beaverton, OR 97005
(503) 644-2686

The Real Oregon
Computer Co.
205 West 10th Ave.
Eugene, OR 97401
(503) 484-1040

Byte Shop Computer Store
2033 SW 4th Ave.
Portland, OR 97201
(503) 223-3496

RHODE ISLAND

Computer Power, Inc.
M24 Airport Mall
1800 Post Rd.
Warwick, RI 02886
(401) 738-4477

SOUTH CAROLINA

Byte Shop
2018 Green Street
Columbia, SC 29205
(803) 771-7824

TENNESSEE

Microproducts & Systems
2307 E. Center St.
Kingsport, TN 37664
(615) 245-8081

TEXAS

Byte Shop
3211 Fondren
Houston, TX 77063
(713) 977-0664

Interactive Computers
7646½ Dashwood Rd.
Houston, TX 77036
(713) 772-5257

The Micro Store
634 So. Central
Expressway
Richardson, TX 75080
(214) 231-1096

VIRGINIA

The Computer Systems
Store
1984 Chain Bridge Rd.
McLean, VA 22101
(703) 821-8333

Media Reactions Inc.
11303 South Shore Dr.
Reston, VA 22090
(703) 471-9330

WASHINGTON

Byte Shop Computer Store
14701 N.E. 20th Ave.
Bellevue, WA 98007
(206) 746-0651

The Retail Computer Store
410 N.E. 72nd
Seattle, WA 98115
(206) 524-4101

WISCONSIN

Madison Computer Store
1910 Monroe St.
Madison, WI 53711
(608) 255-5552

The Milwaukee
Computer Store
6916 W. North Ave.
Milwaukee, WI 53213
(414) 259-9140

CANADA

Trintronics
160 Elgin St.
Place Bell Canada
Ottawa, Ontario K2P 2C4
(613) 236-7767

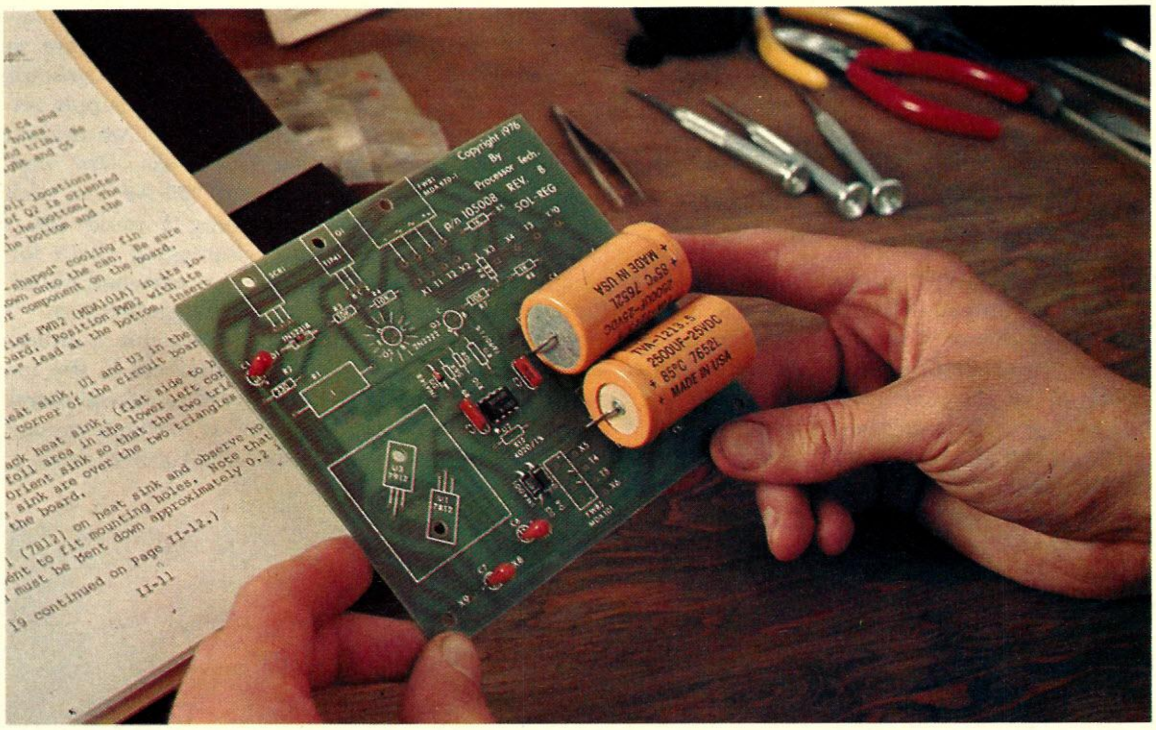
First Canadian
Computer Store Ltd.
44 Eglinton Ave. West
Toronto, Ontario M4R 1A1
(416) 482-8080

The Computer Place
186 Queen St. West
Toronto, Ontario M5V 1Z1
(416) 598-0262

Pacific Computer Store
4509-11 Rupert St.
Vancouver, B.C. V5R 2J4
(604) 438-3282

Processor Technology
Corporation

The Kit and I



Part Two:
or
Power to the Computer

by Richard W. Langer

The first page of my Sol manual said "STOP! DO NOT PASS GO WITHOUT READING THIS PAGE." It then proceeded to tell me where to install the errata sheets. Comforting to know that the people designing these things make mistakes too. . . .

After making the errata sheet insertions, the manual suggested, the best way to go about assembling the Sol was to start with Section II followed by Section III and so on. It didn't say anything about Section I, but I decided to read it anyhow.

The first paragraph of Section I suggested that I scan the whole blankety-blank manual before I did anything else—excluding Sections VII and VIII, I assumed, since the first page of the manual had already informed me that they hadn't finished writing these yet.

Scan, indeed. How was one to scan about five pounds' worth of schematics and instructions? Particularly, how was one to scan paragraphs such as: "Parallel interfacing is eight bits each for input and output plus control handshaking signals, and the output bus is tristated TTL for bidirectional interfaces. The serial interface circuit includes both asynchronous RS-232 and 20 mA current loop provisions, 75 to 9600 baud (switch selectable)"? I was left with, if not the mentioned handshakes, a semirobotic head shake

Photographed by Thomas Hall

from shifting back and forth between text and dictionary, and I was most desirous of discovering why the people in the manuals department couldn't employ someone who communicated in English, or at least english, though preferably both.

After I'd spent half an hour on the first two pages, they started to make a little sense. I found myself at last reading "microamp" instead of "mA." And I now grasped mentally the difference between "serial" and "parallel" without having to use the mnemonics of breakfast and gym bars to try to remember what it was all about. I was learning, though as yet I was far from sure quite *what* I was learning.

690-220-P had no P on the sink itself. Still, one can't have all the Ps and Qs.

Onward to another one of those many parts lists. Unlike the memory board, where the leftovers had to go somewhere on the unit, when it came to the Sol Power Supply (what Section II is all about), the leftovers were the rest of the computer. Logic was a little more difficult to apply in the figuring out of which little doojiggy was which.

There were several bags of parts from which to choose. And on the bags were numbers such as 64, 67, 66—or was it 99? Maybe these referred to the corresponding section numbers? No such luck. Then how about table numbers? Tabula rasa.

Back to the contents approach. One bag had a plug and wire in it, the same as the apparatus connecting my lamp with the wall. Obviously the contents of this bag were to plug a thing into a wall outlet. This bag, then, must have something to do with the Sol Regulator; after all, the section was headed Power Supply. I opened Pandora's plastic bag. Inside, besides the pluggable parts, were a host of baglets filled with some things I'd begun to recognize and numerous other things that as yet remained incognito. Included were some baglets marked Amphenol, which made me wonder if the kit came complete with seemingly soon-to-be-needed barbituates as well as the more directly applicable assembly pieces.

One itemized part I could not fail to recognize immediately was the board onto which everything except the nebulous leftovers would be eventually affixed. It said Sol-REG. Right board, anyway. On to figuring out where everything else would go.

Skipping all the numbers on Table 2-1, I started on Table 2-1-(Continued), since that was where the board was mentioned. The next item listed was a heat sink. Ah, familiar territory once more. Although this heat sink looked quite different from the one used on the memory board, the underlying principle of this function was obviously the same. (My old physics teacher would never have dared to flunk me now.) Besides, there were numbers 690-220-P, to match. Well, almost. Actually there was no P on the sink itself. Still, one can't have all the Ps and Qs. Besides, there was an arrow, which by now I'd learned was a very handy thing to have on a part.

So much for the heat sink. But, as it turned out, there were supposedly three of them. Obviously things around the power supply get hot. Less obvious was the whereabouts of the other heat sinks. One was allegedly numbered 203-AP. No such number on anything I could see. However, while rummaging about for it, I did discover an overshot water wheel of diminutive proportions. This was probably heat sink number 3, which bore no numbers in the manual but was referred to instead merely as "aluminum." It felt light enough.

Once past the heat sinks, I definitely

The coaxial cable attached to the plate by means of the mysterious Amphenol connection.

had to skip about a bit to find things I might recognize, a fuse holder, for instance, or three mica insulators. Mica. I remembered what that looked like from geology class.

The Red-Hots-sized 15 ufd capacitors presented no problem. They were the only things there was supposed to be three of besides the mica insulators. The two 2500 ufd capacitors were something else again. I kept looking for something as small as the ones called for on the memory board. The ones for the Regulator, on the other hand, turned out to be almost as large as 35 mm film cartridges. That size discrepancy, however, guided me unerringly to the 18,000 ufd electrolytic capacitor, which was vaguely hand-grenade size.

Can transistors I again recognized from my past experience with the memory board. They were the parts illustrated in the manual but never to be found in the memory board kit because they'd been replaced since the original design. A third transistor threw me for a loop; it looked just like the diodes I'd culled out before. One must never judge a transistor by its cover, obviously.

Floundering about among the remaining unknowns, I came across the listing "4 Tie Wraps." Now here was a bit of technological overlap of the kind that occurs frequently, and in this case fortuitously, in our consumer society. We use similar devices to close off the plastic garbage bags preferred by our local dumps. That use combined with the descriptive nomenclature gave rise to immediate identification. Stopping while I was ahead, I called it quits for the night. Before I could proceed much further, I'd have to get something to indicate what the color code on resistors meant. There were a dozen of them, all listed by ohms in the manual but themselves marked with nothing besides their color codes.

The 4 Tie Wraps looked like the devices preferred by our local dump for garbage bags.

Comparing boards, I discovered during my next bout with the parts list, helped to define some of the kit ingredients. An old friend, IN 4001, was immediately recognizable from the memory board. Out of the corner of my eye I also spotted a 39-ohm resistor residing in another bag. I'd used that one before too. And once I started to assemble the fan closure, the picture became much clearer, for the instructions referred me to some exploded schematics in Section X. Some commoning blocks I had isolated in the pile of parts by the process of elimination turned out to be just what I'd guessed they were. So, for that matter, did almost everything else.

Q2 is, and will remain, the only piece on my whole board that's crooked.

Part of the problem with wading through a whole instruction book of this kind before starting to assemble or even sort out the parts is that it's much easier for a novice like me to understand things if I can finger the parts as I go. Still, reading the manual through first—maybe several times—would no doubt save the more experienced kit builder a lot of assembly time.

The first step in putting together the fan closure was to screw the fan and guard to the plate with four 6-32 by ½ inch binder or pan head screws. The parts list only mentioned one. I found a bag of extra screws and pilfered, figuring that if I needed more later I could always try to find some at the hardware store. The remedy turned out to be not far removed from the necessities of the situation. The same screws were needed for the AC power cord receptacle; these I pilfered from yet another bag. And so on the list for my next weekly Wednesday trip to the open-Wednesdays-and-Saturdays-only dump, the grocery store, and the hardware store, I put a box of 100 pan head screws.

Once I had the on/off button in place, I stuck a piece of masking tape over it so it wouldn't get scratched as I banged the plate around on the table trying to get the nuts screwed in around the tight spots. That accomplishment required the manual dexterity, not to mention the narrow fingers, of a ten-year-old.

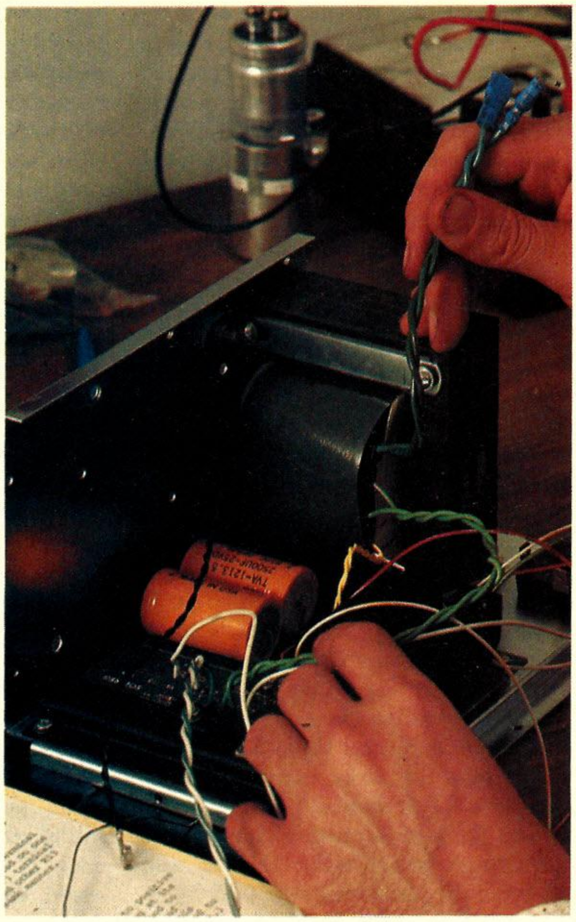
Fuse holder, power cord receptacle, and some other parts were to be attached next. That done, the job at hand was to tackle the female coaxial connector. With the aid of the recently discovered charts in Section X, I located same as one of the mysterious Amphenols I'd wondered about previously—the Amphenol connection redux.

When everything had been attached to the plate, progress became a matter of connecting all the pieces. This entailed use of a piece of coaxial cable which resided in the Sol-PC kit. Aha, so all the pieces didn't come in the same bag. My relief in finding that my previous pirating might have been quite legitimate was very soon overshadowed by the task of cutting off a thirteen-inch section of cable, a task to which my diagonal cutters were not equal. Luckily my pruning shears were still in my back pocket from an afternoon of trimming the suckers off an apple tree. They accomplished the deed.

Stripping the required inch of insulation posed no real problem; rebraiding the shield into a single lead was something else again. I did it about as swiftly and skillfully as my daughter Genevieve does her hair in the morning before setting out for kindergarten. As it turned out, however—I should have reread instead of redoing—while the wire was supposed to be unbraided, it did not have to be braided together again. Simply twisting it together, a much simpler task, would have sufficed.

The lockwasher lugs pictured and specified in the manual did not seem to exist. I found a pair of plain ones and settled for those. The last couple of pieces of the unit snapped into place, and I was most pleased to follow the next instruction: "Put the fan closure assembly to one side." Time for a cup of cappuccino. I'd have made it two except that I'd need steady hands for the Sol-REG board.

I still hadn't figured out all the resistors for the Regulator and was puzzling over how to solve this problem when, lo and behold, as I turned the page this time, I found the color code



Twisting the appropriate wires together was at least a little simpler than braiding my daughter's hair with two minutes left for her to catch the school bus.

for the resistors as well as their actual values. Now why couldn't they have put those in with the parts list?

Actually I'd long since made up my mind to install the capacitors first. There were fewer of them, and surely that bit of deviation from instructions shouldn't do much harm. The only problem was that what I assumed to be the .1 disc capacitors weren't round. They were rectangular. Still, they were made of the right material, and they were marked .1 m. It was as close an identification as I could get. So I installed them.

I had been particularly looking forward to the 2500 ufd capacitors. They were bright orange and almost the size of 35 mm film containers. A nice break in pace.

I went outside to hoe the watermelon patch, muttering a few things to myself about a remedial reading course.

With the capacitors in place, there was no avoiding the resistors any longer. I returned to them. Then I approached the can transistors, with which I hadn't been familiar previously. Both "familiar" and "previously" are terms used very relatively; the first means I now recognize a can transistor, well, most of the time, the latter means PS, as in pre-Sol rather than postscript.

The Q2 can transistor was supposed to be capped off with a star-shaped heat sink. Q2 is, and will remain, the only piece on my whole board that's crooked. The tipped bowler hat offends my aesthetic sense, but I'm not about to try to resolder an old Q2 once I've clipped the leads.

Speaking of soldering difficulties, attaching something called a bridge rectifier presented me with a real problem. The solder just wouldn't flow right. Hope I didn't make things too warm for it.

U1 and U3 were supposed to be installed inside a large heat sink. The U1 went in fine. On the U3 I bent the leads so it went in upside down. That I could tell because of the recessed hole for the screw. For a moment I panicked over metal fatigue, but the leads bent back and over without breaking. What I should have panicked over was that old manual dexterity again. Getting those three leads in place deep down in the sink was like trying to fish the old proverbial ring out of the proverbial kitchen sink trap without taking it apart.

Nevertheless, by the end of fifteen minutes of jiggling, U1 was in place and so was U3. Being more or less the same parts, one numbered 7812 and the other 7912, I hadn't given much thought to installing the second of the two in the same manner as the first. Hah! The second required a piece of mica under it to separate the IC from the heat sink. And then, since there were two identical pieces of mica around, I figured they'd forgotten to mention it for U1, and I might as well put both where they belonged. It was after I'd reassembled the whole thing again that I noticed the other, similar transistor hanging on the wall. Hastily skipping ahead in the instructions, I discovered, of course, that it was the item for which the second piece of mica was supposed to be reserved. I went outside to hoe the watermelon patch, muttering a few things to myself about a remedial reading course.

Returning from the garden some time later, I took out the mica and reassembled everything once more. My dexterity was definitely improving, at least on this particular routine. Maybe I could get a job on the Processor Tech assembly line.

Q1, the transistor requiring the second piece of mica, came next. It must be really hot stuff, because it needed not only the mica and a heat sink, but a nylon screw as well.

That something called a bridge rectifier which so tried my soldering skills was supposed to sit on the same heat sink. A cautionary note indicated that the plus lead must be on the right. (Twice that was mentioned.) By now I'd begun to double-check everything, and I was confident enough that my right would be their left. But, just to be sure, I undid the previous piece and lifted up the heat sink to check the writing underneath. Sure enough, the plus sign was on my left. I'd turned the

Just checking it all out to see if there really was enough room left for the hand-grenade-sized capacitor.

board around and was about to install the piece backwards.

> *By now I'd begun to double-check everything. Sure enough, I'd turned the board around and was about to install the piece backwards.*

When all was done, frontwards, I had another fling with the ohm meter. I think everything tested out all right. But I definitely need some reading matter that goes into greater detail on the operation of this instrument and the various scales.

The transformer which was to be installed on the chassis after dinner weighed about a ton and a half—or at least enough to shorten the life span of the nail on my index finger by several months. Dropping the transformer and chassis on the same said finger while attempting to fit the retaining screws from the bottom as the instructions specified, I crunched the nail. By the next day it had turned several shades darker than Sol blue.

Braiding time came around again. It seemed the black wires should be twisted together, the green wires should be twisted together, the yellow wires should be twisted together, the blue wires should be twisted together .... They got stuck on the white, however. There was only one white wire. Why they couldn't have installed double wires or pretwisted ones still has me baffled. So, for that matter, does the fact that they had to be twisted at all. I know neatness counts, but.

Step 26 called for a four-wire cable. No such animal in the kit. There was a five-wire cable, but that turned out to be for step 27. It was 10:00 at night on a Sunday that I reached steps 26 and 27, so there wasn't much of a chance that there'd be someone at Processor to solve my problem. I finished off step 27 and went fishing.

Two smallmouthed bass later (Mepps number 2 Black Fury, slow jerky retrieve by the yellow waterlilies on the slow-water side of the pond), and I'd figured out that the cable was probably in some other part of the kit. Say like where it plugged in. I rummaged and found one whose wires matched the color coding called for in the instructions. I decided to take another chance. The rest of the evening was spent hooking up the two hand-grenade-sized capacitors and attaching wires into a colorful dish of spaghetti. Then at last the Sol Power Supply was completed. By the time I turned out the lights over my worktable, I was beginning to entertain a definite foreboding that everything would not be in working order when I could finally plug it in. Still, building the kit is turning into something more addicting than potato chips. On to the brains themselves. ▼

HOW COMPUTERS WORK

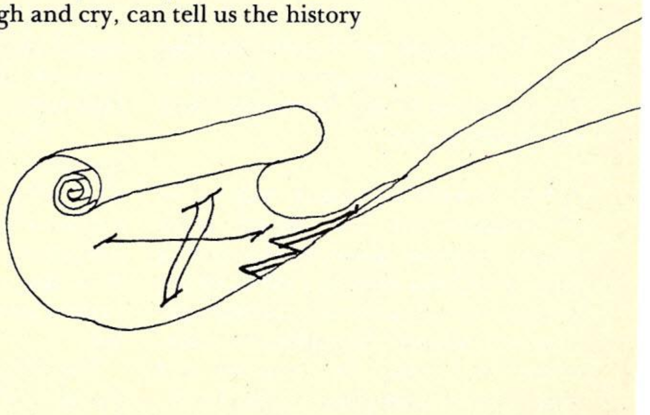
by Joseph Weizenbaum

As seen from one strictly formal point of view, modern computers are simply Turing machines that operate on an alphabet consisting of the two symbols "0" and "1" and that are capable of taking on an astronomical number of states. But this is like saying that, because both bicycles and modern passenger aircraft are vehicles for transporting people, they are formally identical. The modern computer differs from the Turing machines we have been discussing both in the way it is constructed and the way it is instructed.

Many people know that a computer can compare their names as imprinted on credit cards with names somehow stored inside the computer. Yet most people believe computers are fundamentally machines that can do arithmetic on a grand scale, i.e., that they are merely very fast automatic desk calculators. Although this belief is defensible on strictly formal grounds, it is much more useful to recognize that a computer is fundamentally a symbol manipulator. Among the symbols it can manipulate are some that humans, and in a certain sense even computers, interpret as numbers. Still, most computers spend much, even most, of their time doing nonnumerical work.

To justify what I have just said, I must say something about symbols and their interpretation. And, in order to do that, I must also explain how symbols may be represented, especially inside computers.

Let us, at least for now, restrict our attention to symbols used to compose text. These are the upper-case and lower-case letters of the English alphabet, punctuation marks, and such special symbols as those used in mathematics, for example, parentheses, and addition and equality signs. The blank (or space) also counts as a distinct symbol. Were that not so, we would have a hard time writing sentences composed of individual words. Books composed of strings of only these symbols can make us laugh and cry, can tell us the history



From *Computer Power and Human Reason,* Copyright © 1976 by W. H. Freeman and Co.

Illustrated by Cindy Hain

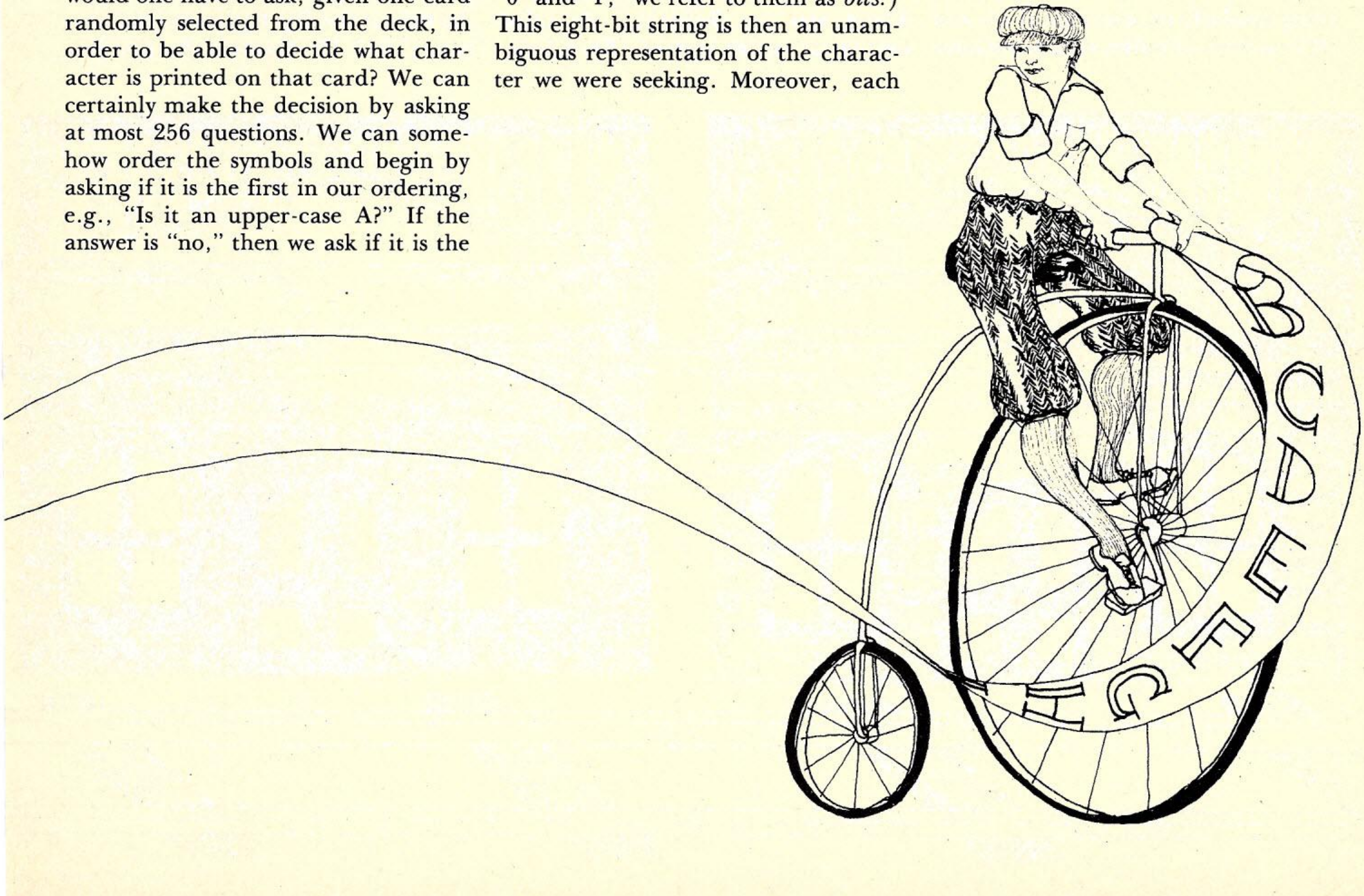
of philosophy, of an individual, or of a nation, and can instruct us in many diverse arts, including that of mathematics. In particular, they may teach us how to construct algorithms, i.e., effective procedures, and they may also give us sets of rules that constitute algorithms. Thus, however informal a notion of what information is we may appeal to, we must agree that the symbols we mean to discuss here are capable of carrying information. How are symbols represented and manipulated in computers?

Computers spend most of their time doing nonnumerical work.

Suppose that the alphabet with which we wish to concern ourselves consists of 256 distinct symbols, surely enough to include all the symbols to which I have alluded. Imagine that we have a deck of 256 cards, each of which has a distinct symbol of our alphabet printed on it, and, of course, such that there corresponds one card to each symbol. How many questions that can be answered "yes" or "no" would one have to ask, given one card randomly selected from the deck, in order to be able to decide what character is printed on that card? We can certainly make the decision by asking at most 256 questions. We can somehow order the symbols and begin by asking if it is the first in our ordering, e.g., "Is it an upper-case A?" If the answer is "no," then we ask if it is the second, and so on. But if our ordering is known both to ourselves and to our respondent, there is a much more economical way of organizing our questioning. We ask whether the character we are seeking is in the first half of the set. Whatever the answer, we will have isolated a set of 128 characters among which the character we seek resides. We again ask whether it is in the first half of that smaller set, and so on. Proceeding in this way, we are bound to discover what character is printed on the selected card by asking exactly eight questions. We could have recorded the answers we received to our questions by writing "1" whenever the answer was "yes" and "0" whenever it was "no." That record would then consist of eight so-called *bits* each of which is either "1" or "0." (When speaking in terms of decimal notation for numbers, we refer to the numbers 0, 1, ... , 9 as *digits*. But our notation permits us only the two symbols "0" and "1;" we refer to them as *bits.*) This eight-bit string is then an unambiguous representation of the character we were seeking. Moreover, each character of the whole set has a unique eight-bit representation within the same ordering.

There do, in fact, exist widely agreed upon conventions for ordering just such a set of characters and for their individual encodings. (That these conventions are not universally agreed to need not concern us here, at least not for the moment.) In recent years the specific coding scheme used in computers manufactured by the IBM company has become very nearly a worldwide industry standard. Within that convention an eight-bit string representing a character (of a 256-character alphabet) is called a *byte* and a chain of four *bytes* a *word.*

We have seen that any text can be represented as a string of 1's and 0's. To do any useful work on information encoded as bit strings, we must be able to manipulate them in some orderly way, i.e., to play games with them. We now know the pieces of the games we may wish to play. All that remains is to state rules. But before we come to



that, let me say a few words about the electrical representation and manipulation of bit strings.

We may say about any wire that an electric current is flowing in it or not. Consider a wire connected to a source of electric power and a suitably connected switch. When the switch is closed, current flows through the wire, otherwise not. Suppose that the switch is connected to a mechanism that opens and closes it regularly; say, the switch is closed for one second, then open for one second, and so on. We may then speak of the flow of electricity on the wire as a pulse train (see Figure 1). An ordinary electric doorbell is a pulse generator. When power is supplied to it, i.e., when the bell button is pushed, a switch is closed. Current flowing through a wire causes the bell's hammer to move, opening the switch. The switch is then again closed, and so on.

A modern computer is, of course, fundamentally an electrical device, just as an electric doorbell is. When we push a bell button, we think of the bell as being "on," even though it, in a sense, turns itself on and off while the button remains depressed. An operating computer may be thought of as being similarly on and turned on and off by a train of pulses such as we have discussed. Conceptually, then, a computer's time is divided into two kinds of intervals, a quiescent interval during which it is, in a sense, "off," and an active interval during which anything that is to happen must happen. In effect, the regular pulse train we have discussed acts as a clock. A state of "no current" on the wire carrying that train signals the quiescent period, and current on it signals the active period.

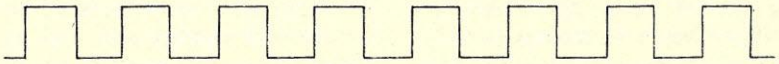
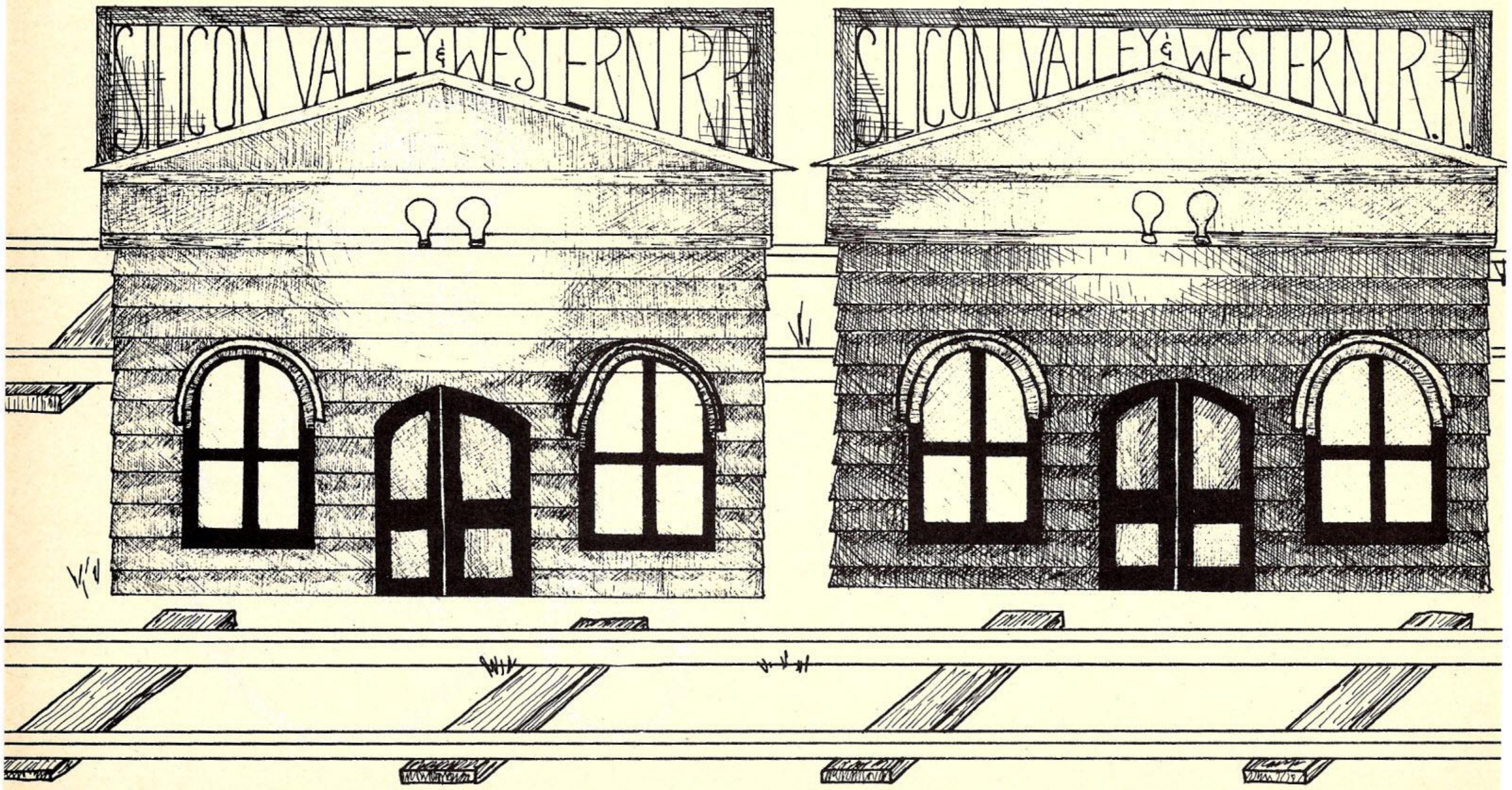


Figure 1. A pulse train of square waves.

To conceptualize what goes on inside a computer, think of a railway map, representing the entire rail network of a continent, in which the actual rail lines are represented by wires. Each railway station is represented by a pair of little neon bulbs. When we look at this network of wires and bulbs during a quiescent interval, we notice that some bulbs are lit and others are dark. During the next active period some of the lit bulbs go off, some dark ones go on, and some remain as they were. That's all. Then there is another quiescent period. There was no flickering of bulbs during the active interval. Each bulb either remained as it was or changed its state exactly once, i.e., turned off if it was on or turned on if it was off.

The device we have conceptualized as a pair of neon bulbs is an electronic circuit consisting of two identical components. Each component is capable of circulating an electric current indefinitely, i.e., of in effect *holding* a pulse. However, only one of the pair may hold a current at any one time. Two wires lead into the device, one to each component. When, during an active period, a wire transmits a pulse to a component that is not then circulating a current, a current is induced in it and the current circulating in the other half of the device is shut off. This device is thus able to flip and flop between two states; either one of its halves is "on" and the other "off" or vice versa. It is therefore called a *flip-flop*. Each of its components also has a wire emanating from it, and each wire will, of course, carry a current, i.e., a pulse, during an active period when the half of the flip-flop corresponding to it is "on." The function of a flip-flop



SILICON VALLEY & WESTERN R.R.
SILICON VALLEY & WESTERN R.R.

in a computer circuit is to "remember" on which of its two sides a pulse last impinged. It is a one-bit information-storage device.

Whenever a commercial begins, they all rush to their telephones and shout.

We now have another way of saying what goes on inside a computer during an active period: many flip-flops change state. But the function of a computer is to manipulate information, not merely to transmit it from place to place. And information manipulation is, as we have already observed, fundamentally a matter of transformation. Now any single wire leading from a side of one flip-flop to that of another can carry at most one pulse during a single active interval. Therefore, whatever transformations are to be achieved during an active interval must be results of electrical operations, not on streams of pulses following one another in time, but on a line of pulses advancing in parallel.

Suppose there were an enormous telephone network in which each telephone is permanently connected to a number of other telephones; there are no sets with dials. All subscribers constantly watch the same channel on television, and whenever a commercial, i.e., an active interval, begins, they all rush to their telephones and shout either "one" or "zero," depending on what is written on a notepad attached to their apparatus. They also listen for what is being shouted at them, and write either "one" or "zero" on their pads, depending on what they hear. A telephone may be a transmitter to several receivers at once, and it may also be a receiver for more than one transmitter. Yet the signal reaching each receiver must be an unambiguous "one" or "zero." There must therefore be operators (actually, electronic devices), placed along the wires connecting receivers to one another, whose function is to compose a single signal, "one" or "zero," from possibly many incoming signals. These devices are called gates.

Let us describe three different kinds of gates, each with a distinct function. The simplest is one that, when it receives a "zero" transmits a "one" and vice versa. This is the NOT gate. Its function is described by the formulas

$$\mathrm{NOT}(0) = 1,$$
$$\mathrm{NOT}(1) = 0.$$

A common schematic representation for it is shown in Figure 2. It has one input, here labeled A, and one output, here labeled B.

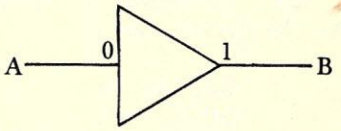
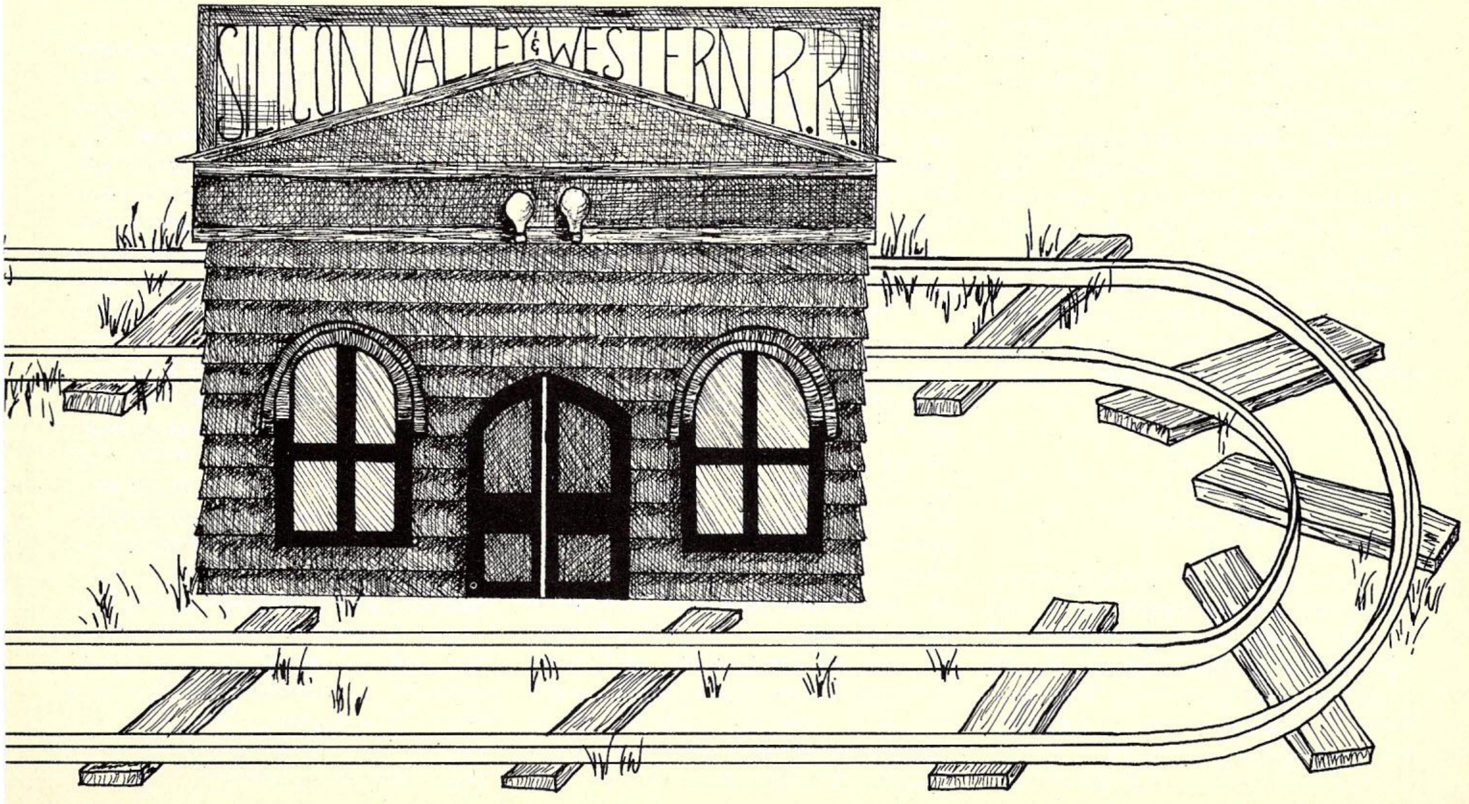


Figure 2. A NOT gate.

The AND gate has two inputs and one output. It transmits "one" if and only if its two inputs are both "one;" otherwise it transmits "zero." Its function is described by the formulas

$$\mathrm{AND}(0,0) = 0,$$
$$\mathrm{AND}(0,1) = 0,$$
$$\mathrm{AND}(1,0) = 0,$$
$$\mathrm{AND}(1,1) = 1.$$

Its common schematic representation is shown in Figure 3.


SILICON VALLEY & WESTERN R.R.

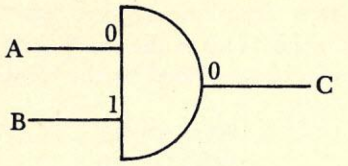


Figure 3. An AND gate.

The OR gate also has two inputs and one output. It transmits "one" whenever either or both of its inputs are "one;" otherwise it transmits "zero." Its formulas are

$$\mathrm{OR}(0,0) = 0,$$
$$\mathrm{OR}(0,1) = 1,$$
$$\mathrm{OR}(1,0) = 1,$$
$$\mathrm{OR}(1,1) = 1.$$

Its common schematic representation is shown in Figure 4.

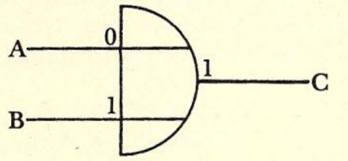


Figure 4. An OR gate.

The highly motivated reader may wish to trace pulses through the system of gates shown in Figure 5, which represents a circuit whose components are AND, OR, and NOT gates and whose function is to add two binary digits arithmetically.

(Binary addition is like decimal addition, only much simpler. The decimal sum of 2 and 3 is 5. The decimal sum of 7 and 8 is also 5 but there is also a "carry" of 1, which is added to whatever the sum of the next column to the left is. In binary arithmetic, there are only two digits, 0 and 1. The addition rules are very simple, namely,

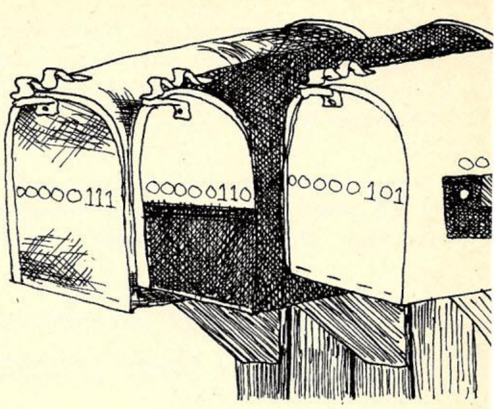
$$0+0=0,$$
$$0+1=1,$$
$$1+0=1,$$
$$1+1=0 \text{ and carry } 1.$$

The one-bit binary adder shown therefore has three inputs, *A, B,* and *C,* and two outputs, *S* and *D*. *A* and *B* represent the two bits to be added, and *C* any carry that may have been produced by a similar adder to the right, so to say, of the one shown here. *S* is the sum produced, and *D* the carry.)

It is really not necessary for the reader to trace pulses going through this adder. The important thing is to understand that combinations of the three simple gates we have described are capable of realizing transformation rules on information. Notice also that we enclosed the whole circuit in a box. This box has three inputs (*A, B,* and *C*) and two outputs (*S* and *D*), and is itself a unit. We have combined simple components to make a more complex component. We could now combine, say, thirty-two of these adders and form a thirty-two-bit adder. And that would then be a single component. Both the construction and the instruction of computers is just such a process of making bigger things out of smaller things.

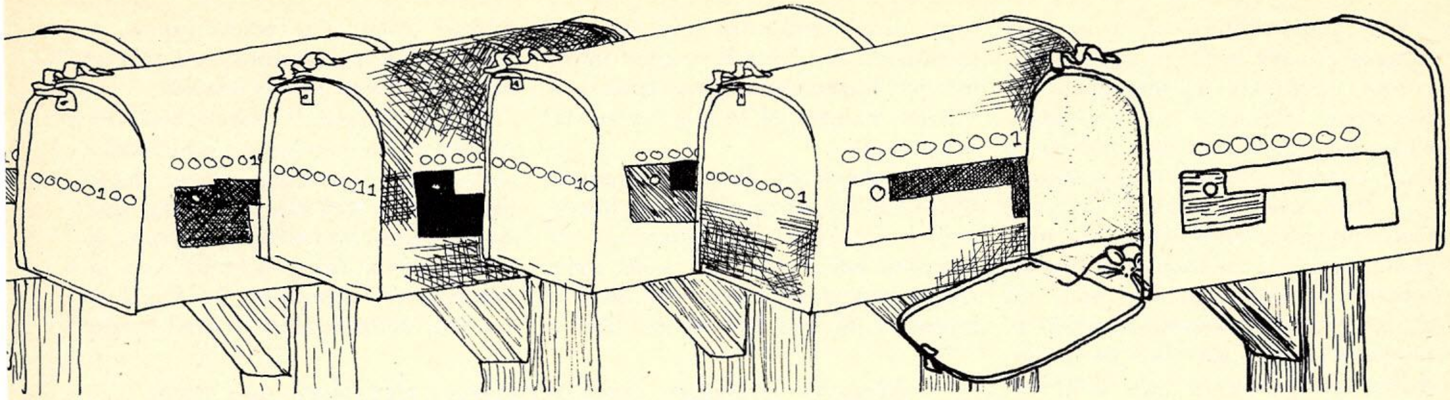
Let us return for a moment to our image of interconnected telephones, but this time with the realization that gates intervene in conversations (if we may call them that) among subscribers. We can now imagine the three subscribers *A, B,* and *C* picking up their telephones at the appropriate time and shouting "1," "0," and "0," respectively. If they are connected to the subscribers *S* and *D* by the circuit we have shown, the *S* will hear "1" and *D* will hear "0," and each will write what he hears on his notepad. Of course, in real computers all these "subscribers" are flip-flops, and the broadcast periods are extremely short. In computers, then, results of computations, i.e., of information transformations, performed during active periods are stored in flip-flops. There they survive quiescent periods in order to become available for further transformation during subsequent active periods.

I have already suggested that one-bit adders may be combined to form, say, thirty-two-bit adders. The inputs to such an adder would be two sets of thirty-two flip-flops each—the carries are internal to a multistage adder—and the output again a set of thirty-two flip-flops. Each of these sets may be said to contain a thirty-two-bit binary number during quiescent periods. Each would of course have to be an ordered set, i.e., one in which there is a first bit, a second bit, and so on. Such a set of flip-flops is called a *register.* It is another example of an aggregation of more elementary components. But may we say about any n-bit register that it contains an n-bit number? No, at least not unconditionally; for what interpretation may be placed on a bit string residing in a register depends on what components are wired to that register, hence what operations may be performed on its contents. If a particular pair of registers is connected only to components that perform arithmetic operations, i.e., if the computer treats the information stored in them only as numbers, then they are numbers—at least while they are being manipulated within the computer. To appreciate that the symbols that occur in natural language are subject to similar constraints, one need only consider this very sentence, in which the word "one" is a number in one place, a word in another, and an uninterpreted character string in a third.



An operating computer is engaged in playing an elaborate and very complicated game. After each quiescent interval it makes a move. The contents of many registers are transported to other registers much as chess pieces are moved over the chess board. But in the computer a great many pieces, bits, are moved at once and, what is most important, individual bits and sets of bits are transformed while on their journey from place to place.

I have already mentioned the fact that in many computers sets of eight bits are aggregated into bytes. An eight-bit register is a physical embodiment of this conceptual aggregation. Computers contain many registers that are connected to one another by combinatorial gating networks of the kind I have described. The entire set of these registers and of the logical networks that unite them constitutes, in a sense, the computer. This set is the machine, often called the computer's central processing unit (CPU), that



actually performs logical symbol manipulation during each of the computer's active periods. The total configuration of states of its individual flip-flops constitutes the state of the computer in a sense strongly analogous to what I meant when earlier I spoke of the state of a Turing machine.

A Turing machine of the kind we discussed earlier gets the information on which it is to operate from a tape. It must read and write this tape sequentially, one symbol at a time. A modern computer, on the other hand, stores much of the information it manipulates internally. The computer's internal-storage device consists of a very large array of eight-bit registers, possibly a million or more of them. These are arranged in a definite order, somewhat like the mailboxes in the lobby of a large apartment house. The registers are numbered serially, beginning with 0, 1, and so on. A register's number is called its *address.* Since each register has a unique address, one can speak of a register's address quite independently of its contents, i.e., the state of its eight flip-flops, and vice versa. Now imagine an array of apartmenthouse mailboxes that is equipped with exactly one combination lock. In order to take anything from a specific mailbox or put anything into it, one must first set that mailbox's address into the lock. The computer's store has just such a device. It is, of course, a register. (A twenty-bit register would be sufficient to address 1,048,576 boxes.) It is useful to think of this address register as part of the computer's CPU, even though it is connected to both the computer's store and its CPU. But to see it as part of the latter helps to visualize that its contents are themselves machine-manipulable. They may, for example, be intermediate results in some long chain of arithmetic computation and may be used as, in effect, ordinal numbers.

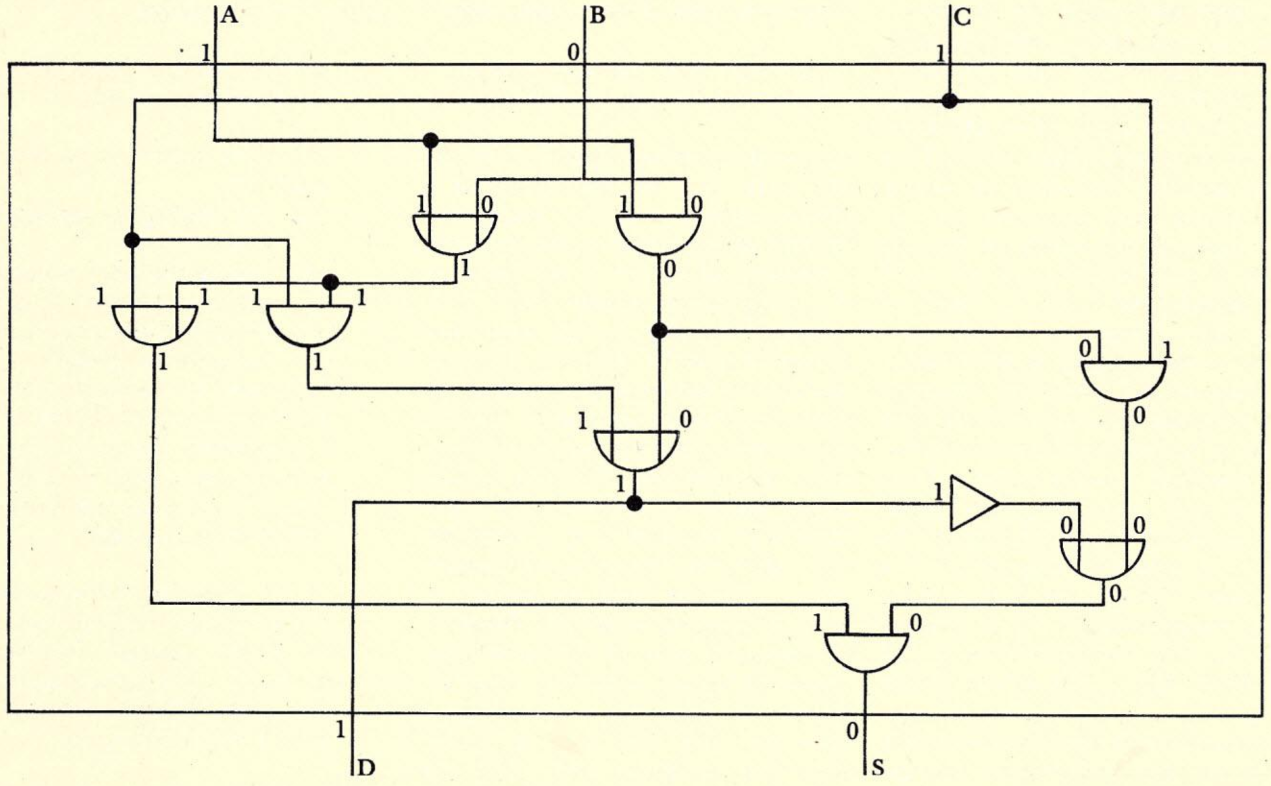


Figure 5. A one-bit adder with carry-in and carry-out.

The addressability of a modern computer's store is one of its most important properties. To appreciate the enormous difference addressability makes just to the way one searches a store, consider the following problem. A certain town has fewer that ten thousand phone subscribers. The telephone directory for that town lists all subscribers alphabetically and, of course, gives their respective telephone numbers. But it also lists each subscriber's serial position in the directory. For example:

1 Aaban, John 369-6244
.
.
.
1423 Jones, William 369-0043

We wish to know whether the last four digits of any subscriber's telephone number are the serial number of any other subscriber whose telephone number similarly corresponds to the first subscriber's serial number. Given the listing shown for Jones above, for example, is there a listing of, say,

43 Baker, Max 369-1423

which would meet these conditions?

A simple way to solve the problem is to look at each listing beginning with the first, and see if it and the listing with the serial number corresponding to the last four digits of its telephone number constitute a pair of the kind we are seeking. The answer is "yes" if we find one such pair, and "no" if, after inspecting the whole set, we find no such pair. The worst case we could have encountered is that in which no pairs exist. Then we would have looked at every listing at most twice. But suppose the telephone directory were recorded on a tape of the kind required by the Turing machines we have described. Then, even apart from tape motion associated with bookkeeping functions, every listing would need to be scanned many more times than twice. The very first listing in our example would require that we look at it, at the 6242 listings between it and the 6244th listing, and at the 6244th listing itself. In classic Turing machines such a tape would not be merely passed over between relevant listings, but each listing would actually have to be inspected and interpreted. The same principle accounts for the anger some people feel when told they are mentioned in a large book that unfortunately has no index. They must then face the prospect of having to read the whole book.

This telephone directory example illustrates not so much that addressability increases the efficiency of searches, but that it can, under some circumstances, help avoid the need to search at all. For had we specified a particular subscriber and asked whether he is paired with another in the way we indicated, that question could have been answered directly and without searching through irrelevant data. Is Mr. Aaban so paired, for example? To find out we look directly at subscriber number 6244. If the last four digits of his telephone number are "0001," then yes; otherwise no.

Many public washrooms display a sign urging users to leave the room as they found it, but we adopt the opposite convention.

With this in mind, let us look back at the quintuples that define some Turing machine *T* and thus constitute a program for a universal Turing machine that is to imitate *T*. Recall that the general form of such a quintuple is

(present state, present symbol,
next state, new symbol, direction).

In general, a Turing machine in a certain state, say, 19, and scanning a certain symbol, say, "1," must read through all quintuples in its program, constantly asking, so to say, whether the particular quintuple (rule) it is currently reading is the one that applies to state 19. When it finds the one appropriate to its state and to the symbol it is then scanning, it rewrites that symbol, moves its tape, and (possibly) changes state. Then the search begins again. If, however, the quintuples were recorded in an addressable store, then the search for the appropriate quintuple could be avoided or at least reduced in length. Suppose, for example, that all the quintuples associated with a particular state required, say, 100 bytes of storage space and that the first one is stored beginning in register 1000. Then the set corresponding to state 19 would be stored beginning in the register numbered $1000+(19-1)\times100$, i.e., register 2800. The whole notion of state is thus transformed into that of address, at least in this context.

In real computers, data too are recorded in the computer's addressable store. This innovation allows us to eliminate as well the restriction that only a datum immediately adjacent to one presently being scanned may be immediately accessible. To begin to see how addressability is used in the composition of real computer programs, let us look at a small but realistic problem.

We want to compute the square roots of numbers. (That is, given a number, say, 25, we want to know what number, when multiplied by itself, will produce the given number. The square root of 25 is 5, because $5\times5=25$.) We assume we have a faithful (human) servant who will tirelessly obey every instruction we give him. We know an algorithm for computing square roots of positive numbers. Given the number *n*, we compute its square-root as follows:

1. First we make a guess, always the same one, namely, *1*.
2. We then arrive at a better guess by:
    (a) multiplying the old guess by itself;
    (b) adding the given number *n* to that product; and
    (c) dividing that sum by twice the old guess.
   (No less an authority than Issac Newton proved that this computation always yields a better guess, unless, of course, the previous guess was already the correct solution.)
3. If the difference between the old guess and the better guess is small enough for our purposes, then we accept the new guess. Otherwise we compute a still better guess.

We justify this procedure as follows. Suppose we wish to find the square root of 25 and we take our initial guess to be, say, 4. We know that this is too small, since $4 \times 4 = 16$. If we divide the number originally given, that is, 25, by our guess, then the quotient will be larger than the result we are seeking, which is 5 in this example. That quotient and the guess we have made therefore bracket the result we seek. If we then take the average of the two, we will get another guess, moreover, one that is closer than the guess on which it was based. We may iterate on this formula until we get a result as close to the correct one as we please.

Our servant, however, is not terribly bright. He has worked with tax forms that require one to calculate this-and-that and to write it on line so-and-so or in box such-and-such. We have therefor made up the worksheet shown in Table 1, namely, one based on such procedures. Our servant works in a little cubicle that has an input slot and an output slot. As soon as a slip of paper with a number written on it appears in the input slot, he begins to work furiously. When he finishes he writes his result on a paper and puts it in the output slot. His only initial instruction is to start by obeying the instruction on line 101 of the worksheet and, unless he encounters an instruction to the contrary on the worksheet itself, to continue obeying the instructions in the sequence in which they are written. He would, by the way, be well-advised to write only in pencil and to have a good eraser at hand; he must use what little space is given him for writing on the worksheet over and over again. (The reader should not attempt to carry out an example computation on the worksheet to the bitter end. He may, however, profit from carrying it out sufficiently far to generate, say, two better guesses for the square root of 25.)

Notice the important role box A plays. It is in effect a register that contains intermediate results of computations. Notice also that no individual instruction refers to more than one line number. Instructions that have this property are called *single-address instructions.* We can perform, say, additions—which, of course, require two operands—by first storing one operand in box A, then adding the other operand to the contents of box A, and leaving the sum again in box A. Lines 121, 123, and 124 serve as temporary storage registers. Finally, notice that the instructions and the storage of intermediate results are so organized that a worksheet once used may be used again. On second and subsequent uses, box A and lines 121, 123, and 124 will contain numbers irrelevant to the new task. But these numbers will not interfere with the newly started computation. In this we followed a quite universally accepted programming practice: whereas many public washrooms display a sign urging users to leave the room as they found it, we adopt just the opposite convention. We say "Put the room in the condition you wish it to be in before you begin serious work." We always store the "old guess," for example, in line 123. After the worksheet has been used, that line will contain the last "old guess" of the last completed computation. But a fresh computation, for which the last stored "old guess" is totally irrelevant, begins by copying "1," our standard first guess, into line 123 (see lines 102 and 103).

Table 1. Directions for Computing Square Roots.

```
101 Get number from input slot and write it on line 121.
102 Copy contents of line 122 into box A.
103 Copy contents of box A onto line 123.
104 Copy contents of line 123 into box A.
105 Add contents of line 123 to contents of box A, and write result in box A.
106 Copy contents of box A onto line 124.
107 Copy contents of line 123 into box A.
108 Multiply contents of box A by contents of line 123, and write result in box A.
109 Add contents of box A to contents of line 121, and write result in box A.
110 Divide contents of box A by contents of line 124, and write result in box A.
111 Copy contents of box A onto line 124.
112 Subtract contents of line 123 from contents of box A, and write result in box A.
113 Copy the absolute value of box A into box A.
114 Subtract contents of line 125 from contents of box A, and write result in box A.
115 If contents of box A are greater than zero, begin work with line 118.
116 Put contents of line 124 in output slot.
117 Stop.
118 Copy contents of line 124 into box A.
119 Copy contents of box A onto line 123.
120 Begin working with line 105.
121 0
122 1.0
123 0
124 0
125 .001
```

Box A

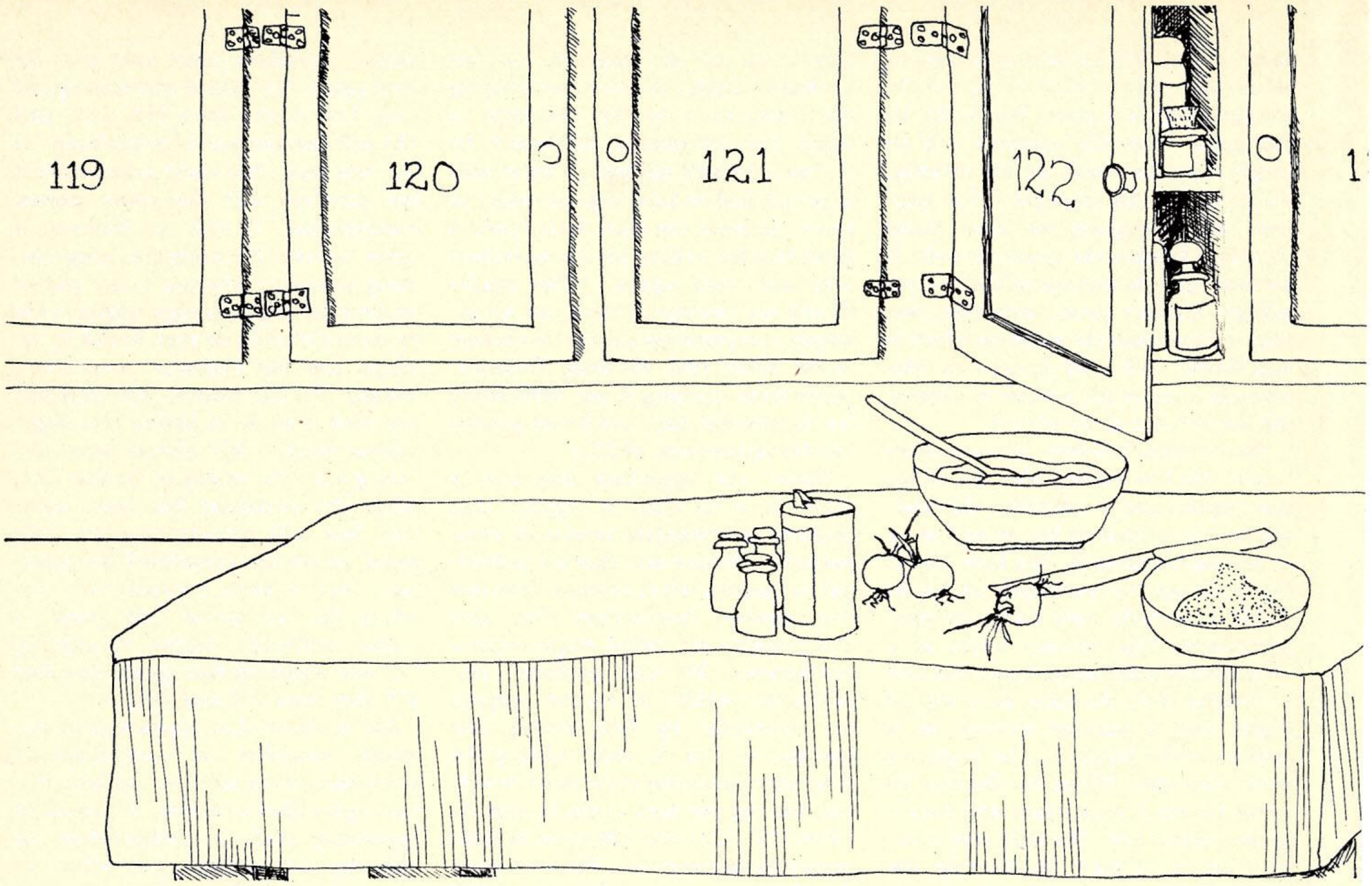
For a reason that appears to be nowhere recorded, a computational procedure of the kind we are here discussing is called a *routine.* We speak of beginning such a computation as entering the routine, and when we have completed the last step, we say we leave the routine. But small routines such as the one illustrated here are seldom of much use in and of themselves; how many times does one really need to know the square root of a number outside of a context established by some larger computational task? Such a larger task may be, say, the computation of the roots of a quadratic equation, i.e., of an equation of the form:

$$ax^2 + bx + c = 0.$$

The desired roots are given by the formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Here then is a problem—i.e., to find the roots of a given quadratic equation—which involves the computation of the square root of some number as a subproblem. A worksheet very similar to Table 1 can be designed that would direct a human computer in the solution of the over-all problem. But, since we already have a worksheet that tells us how to compute square roots, we would be well-advised to tell the

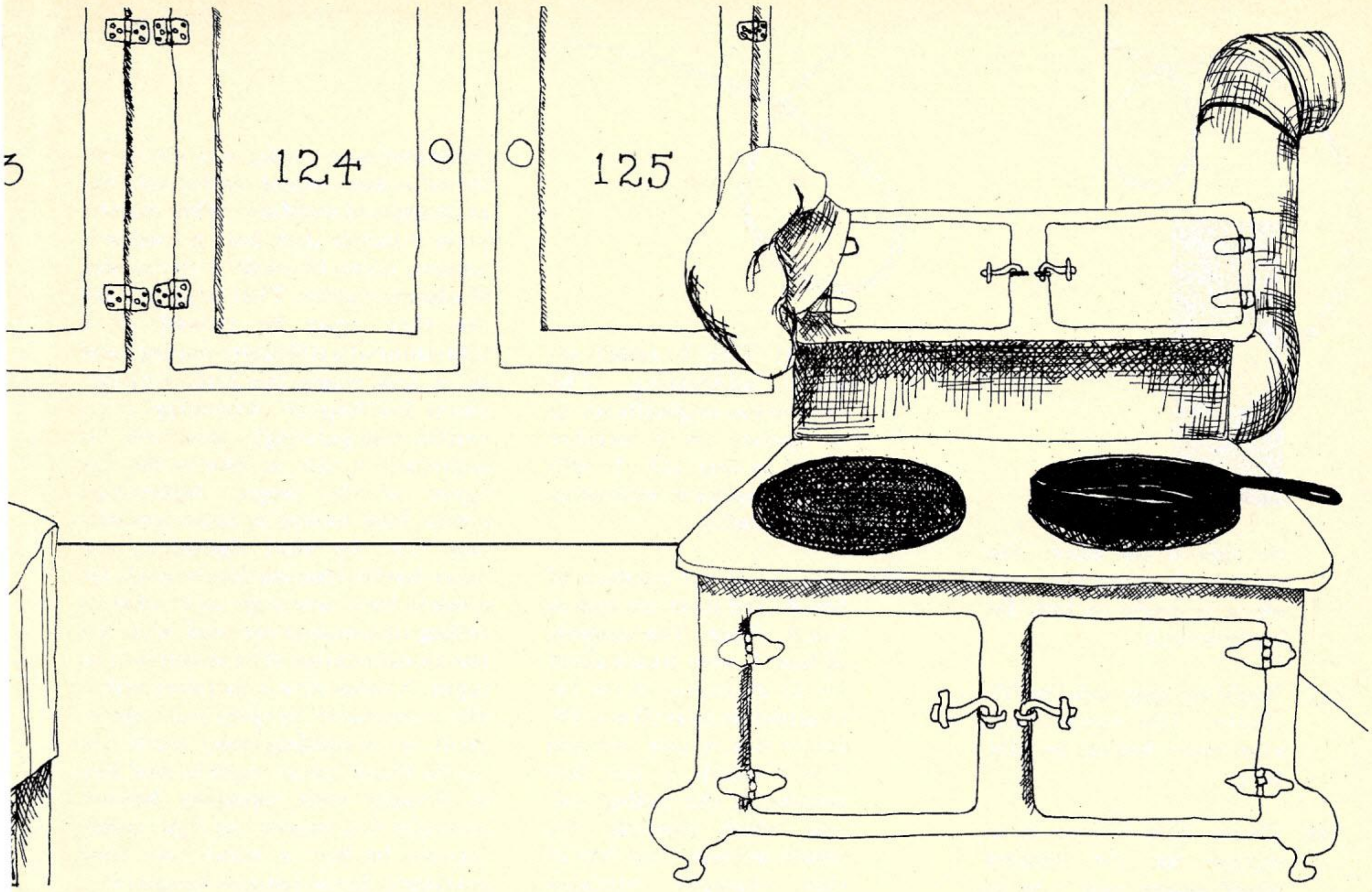


person working on the larger problem to use the already prepared worksheet to help him solve the subproblem when he comes to it, i.e., after he has computed the quantity (b^2-4ac). The square-root routine then becomes a *subroutine* of the larger routine. Of course, that larger routine may again be a subroutine of a still larger routine, and so on. Again I emphasize that much of computers and computation has to do with building larger hierarchical structures out of smaller ones. It is precisely because the square-root routine we have exhibited may play the role of a subroutine in a larger procedure that we have taken care that it initializes itself, that is, puts itself in good order, before each use.

The perceptive reader will have noticed that I have glossed over a management problem that will surely come to haunt us if our computing servants are really as simple, as literal, and as unimaginative in the way they interpret the instructions we give them as I said they were. We have assumed, and we shall stick with the assumption, that a worker knows how to add a number read from a certain line to the contents of box A, how to leave the sum in box A, how to tell whether a number recorded in box A is greater than zero or not, and so on—that, in other words, he knows enough to be able to interpret and to obey all the instructions on the square-root worksheet. But how do we instruct someone similarly trained but working on larger problems to subcontract, so to say, his subproblems?

We can avoid having to confront that question by simply assuming that the worker knows how to compute, say, square roots just as we assume he already knows how to add and subtract. But are we also to assume he already knows how to compute the roots of quadratic equations? If we do not stop making assumptions of this kind we will assume the need for computers away altogether, for then we will have postulated that, in order to compose a worksheet corresponding to any computation whatever, all we have to write is, in effect, "do it." We may be entitled to believe that people can add, subtract, and so on. If we want them to perform more complicated symbol-manipulation tasks, however, we will have to describe appropriate procedures to them in terms of things they already understand. On the other hand, we don't want to have to describe an often-used procedure over and over again. Every routine is potentially a subroutine of some larger routine. Therefore, having once written a particular routine, we would want it to become part of a subroutine library from which it can be called whenever it is needed.

Every library has some convention for calling volumes from the stacks; a reader fills out a request card, hands it to the librarian, etc. But our situation is a little more demanding than that of an ordinary lending library. We want our workers not only to see the worksheet corresponding to the called-for computational task, but also to, in effect, farm out the work itself. For example, suppose a cook has brought a recipe to a certain point and then wishes a specialist to perform the function he is famous for on ingredients the cook has already prepared. The cook may rest while the specialist works, and may continue his labor only after the specialist returns the desired concoction to him. Assuming there are many such specialists, how does the cook turn control of the



kitchen over to just the right specialist, how does he make sure the specialist works on the right ingredients, how does he finally regain control over his kitchen, and where does he find the results of the specialist's labors? One way to do all these things is as follows:

> The cook leaves all the ingredients the specialist is to work on in sequentially numbered cupboards, say, the first in 119, the next in 120, and so on. In the next cupboard after the one containing the last ingredient, he writes the number of the apartment he will be visiting while the specialist works. He leaves the number of the cupboard in which the first ingredient is stored on the stove. Finally, he rings the bell of the specialist's apartment, whose address he, of course, knows.
>
> The specialist knows the cook's conventions very well. When he hears the bell ring, he goes to the kitchen, finds on the stove the number of the cupboard containing the first ingredient, and begins doing his tricks. When he is finished, he leaves his concoction on the stove, rings the bell of the apartment whose address he found in the first cupboard after those that contained ingredients, and leaves.

The specialist knows the cook's conventions. When he hears the bell, he goes to the kitchen and begins doing his tricks.

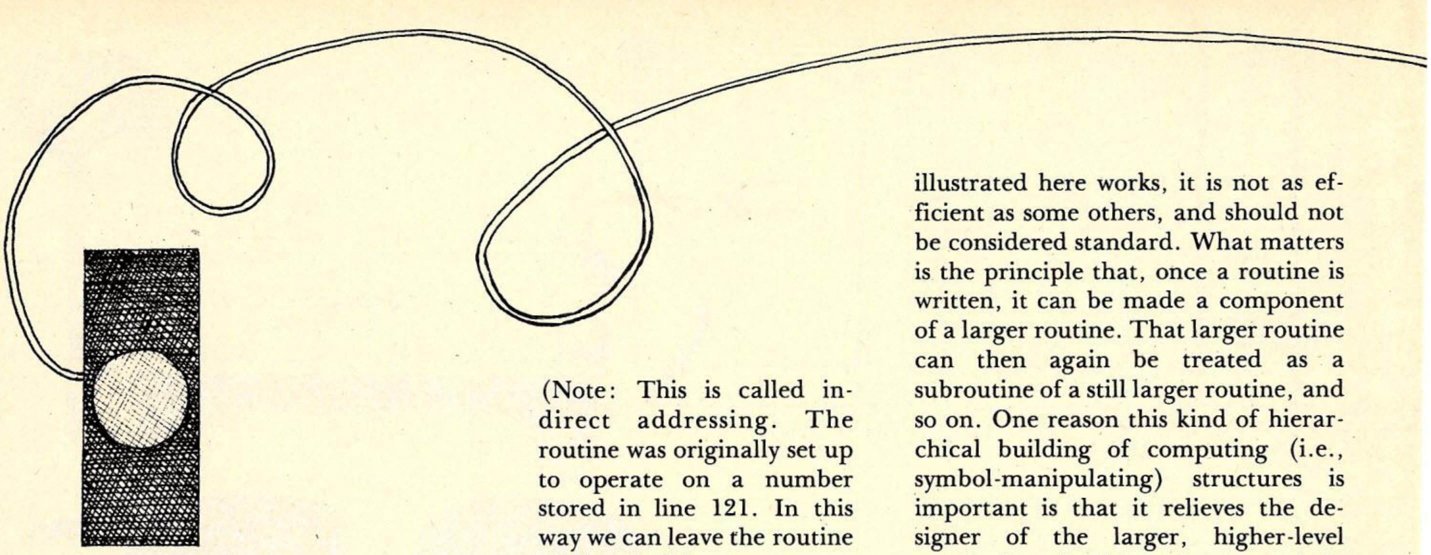
In order for this scheme to work, it is necessary only that the cook and every specialist who may be called on know the conventions, that the cook know exactly how many ingredients the particular specialist he has called needs (he wouldn't want the literal-minded man to boil the note containing the cook's address along with the other ingredients of the stew), and that the cook know the address of the particular specialist he wishes to call.

We can invoke virtually identical conventions for calling subroutines in our worksheet format. Consider, for example, the following fragment of a worksheet. It is part of a larger routine. At the point we take it up, a number whose square root is to be computed is in box A. This fragment calls the square-root subroutine and also assures that control will return to the main routine.

508. Copy contents of line 680 into box A. (Note: Line 680 contains the number whose square root is to be computed.)

509. Copy contents of box A onto line 512. (Note: Line 512 is the "cupboard" in which the first—in this example, the only—"ingredient" is stored.)

510. Copy the number "512" into box A. (Note: Box A plays



the role of the store. The specialist has now been told where to begin to look for ingredients.)

511. Begin working with line 98. (Note: The square-root subroutine begins on line 98.)

512. (Note: This line serves as storage for the number whose square root is to be taken.)

513. 514. (Note: This line contains the address of the line to which the specialist is to return control after he finishes his work.)

514. (Next instruction in larger routine comes here.) (Note: When this instruction is encountered, the square root of the number stored in line 680 is in box A.)

The specialist's worksheet has already been displayed, in Table 1. However, we must modify it to take account of the conventions we are discussing. The original worksheet tells the worker to get the number on which he is to operate from an "input slot." Now, of course, the number of the line containing that datum is in box A. We therefore write:

98. Copy the contents of box A onto line 126.

99. Copy the contents of the line whose number is contained in line 126 onto line 121. (Note: This is called indirect addressing. The routine was originally set up to operate on a number stored in line 121. In this way we can leave the routine undisturbed.)

100. Add "1" to the contents of box A, and leave the sum in box A. (Note: The contents of box A were undisturbed by the execution of the instructions in lines 98 and 99. Hence box A now contains "513," which is the line number of the calling routine which contains the return address; line 513 in this example contains "514.")

101. Copy contents of box A into the address portion of line 117. (Note: Line 117 was a stop instruction but we will replace it as shown below. Since all instructions are single-address instructions, the term "address portion" is unambiguous when applied to instructions.)

We now rewrite lines 116 and 117 as follows:

116. Copy contents of line 124 into box A. (Note: We are leaving the "concoction," i.e., the square root of the number given us, in box A.)

117. Begin working with line 0. (Note: The "0" in this line will, of course, be replaced whenever the instruction in line 101 is obeyed, in other words, each time this subroutine is invoked.)

My aim has been to illustrate one of the many possible ways to call a subroutine. Although the specific way illustrated here works, it is not as efficient as some others, and should not be considered standard. What matters is the principle that, once a routine is written, it can be made a component of a larger routine. That larger routine can then again be treated as a subroutine of a still larger routine, and so on. One reason this kind of hierarchical building of computing (i.e., symbol-manipulating) structures is important is that it relieves the designer of the larger, higher-level system from having to know precisely how (i.e., by what algorithm) the lower-level subroutine does its work. He needs to know how to get at it, what its calling conventions are, and what the functional relation of its output is to its input. A subroutine is therefore rather like a sometimes complex legal instrument, say, a building lease: one fills in all the blank spaces, signs it, and files it. Usually each signatory believes correctly that exactly the legal consequences he had in mind have been entrained. But sometimes one gets into trouble when relying on prepared forms whose "intentions" one thinks one understands and whose fine print one fails to read. I shall have more to say about such things later.

A human worker whose job it is to compose the kind of worksheets I have described would soon tire of having to write so much. He would soon invent abbreviations. No essential information is lost if, for example, the line

(103. Copy contents of line 123 into box A)

of the original worksheet (Table 1) is abbreviated by

(103. GET 123)

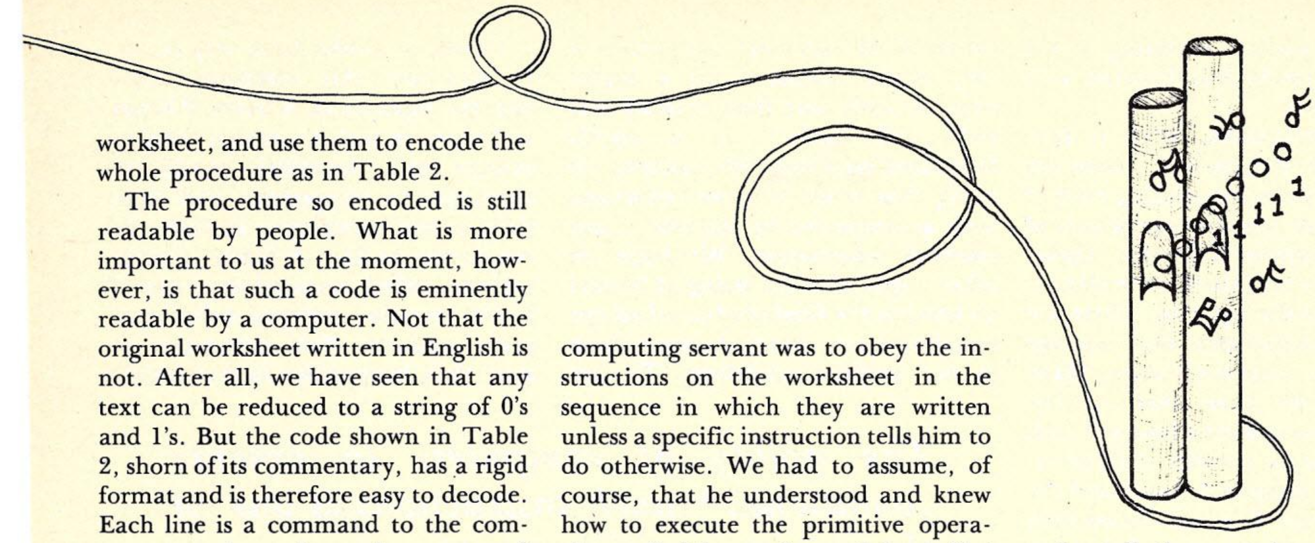
or if the line

(105. Add contents of line 123 to contents of box A, and write result in box A)

is abbreviated by

(105. ADD 123).

We can invent similar abbreviations (in effect, verbs) for each of the operations mentioned in the original



worksheet, and use them to encode the whole procedure as in Table 2.

The procedure so encoded is still readable by people. What is more important to us at the moment, however, is that such a code is eminently readable by a computer. Not that the original worksheet written in English is not. After all, we have seen that any text can be reduced to a string of 0's and 1's. But the code shown in Table 2, shorn of its commentary, has a rigid format and is therefore easy to decode. Each line is a command to the computer, and each such command consists of two components: an operation to be performed, e.g., ADD, and the address of an operand. Box A is no longer mentioned explicitly. It is understood that box A always contains the implicit or unmentioned operand when one is needed, and that the result of any operation, e.g., the sum produced by an addition, is stored there. But from a computer's, as opposed to a human's, point of view, this code is still much too longwinded. The operation-code portion of a command, e.g., ADD, can be replaced by a single byte-sized character. (Recall that there are 256 distinct such characters.) The only restriction such a convention then imposes is that we cannot appeal to more than 256 so-called built-in operations, i.e., operations that require no further explanation to the worker. Such operations are called *primitives.*

To continue the conversion of this example procedure to machine code, suppose that the byte-length code for ADD is "00110101." Then, if the length of a command as stored in a particular computer is four bytes, i.e., thirty-two bits, the actual computer code for line 105 of our worksheet would be

00110101000000000000000001111011.

The whole code for our routine would be an array of twenty-five such lines.

We have illustrated how a computational task may be organized as a procedure executable by humans and how such a procedure may be translated into a notation that, although cryptic, is easily managable by a computer. But we have cheated just a little. Our advice to our faithful computing servant was to obey the instructions on the worksheet in the sequence in which they are written unless a specific instruction tells him to do otherwise. We had to assume, of course, that he understood and knew how to execute the primitive operations called for on the worksheet. But we also tacitly assumed that he could remember his place on the worksheet even while distracted during, say, the performance of a long division. Actually, maintenance of the flow of control in a program is a task quite different and separable from that of executing primitive operations. We should really add another box to the worksheet, namely, a box P (for *program counter*) which at all times contains the line number (i.e., the address) of the instruction then being obeyed. We should then appoint a supervisor whose job is to tell the worker on what line he is to find his instruction, namely, the line whose address is stored in box P. When the worker tells the supervisor that the assigned instruction has been obeyed, the supervisor adds "1" to the contents of box P, stores the new count there, and gives the worker his next assignment. The effect of an instruction that disturbs the normal sequential flow of control is, of course, to replace the contents of box P with the address of the next instruction that is to be obeyed. For example, the effect of obeying the line

(120. Begin working with line 105)

is that the worker copies the number "105" into box P and immediately goes about obeying the instruction written

Table 2. An Encoding of a Square-root Routine.

LINE	COMMAND	COMMENT
101	INP 121	Get number from input and put in 121.
102	GET 122	Get initial guess.
103	STO 123	Store as "old guess" in 123.
104	GET 123	Get "old guess."
105	ADD 123	Double it.
106	STO 124	Store "twice old guess" in 124.
107	GET 123	Get "old guess."
108	MPY 123	Multiply it by itself.
109	ADD 121	Add given number to that.
110	DIV 124	Divide result by "twice old guess."
111	STO 124	Store "new guess" in 124.
112	SUB 123	Subtract "old guess" from "new guess."
113	ABS	Make that result positive.
114	SUB 125	Subtract tolerance from that.
115	JGZ 118	If that is greater than zero, skip next two steps.
116	OUT 124	Put out "new guess."
117	STP	Stop.
118	GET 124	Get "new guess."
119	STO 123	Put it in place of "old guess."
120	JMP 105	Start "loop" again.
121	0	Place for number to be worked on.
122	1.0	Place for initial "old guess."
123	0	Place for "old guess."
124	0	Place for "new guess."
125	.001	Place for tolerance.

on line 105 without reporting to the supervisor that he has finished with one instruction.

In real computers, work is subdivided much further still. There are components for accomplishing each of the built-in or primitive operations of which the computer is capable. These, although they may share some circuitry, are essentially separate. There are still more components that manage access to the computer's store. Many computers even have small special-purpose subcomputers whose sole task is to supervise the transfer of information between the outside world and the main computer. I cannot discuss such details here without being led much too far from my main concerns. I did, however, single out the idea of flow of control of a program, because it is a really important concept. Notice that two lines are drawn to the side of the program shown in Table 2. These illustrate the flow of control in that program, or, to put it another way, the structure of that program. What makes even this little program at all interesting is that it involves a *conditional branch.* Whether or not the program, while running, goes through another iteration is determined by the outcome of the instruction of line 115. If, when that instruction is encountered, box A contains a number greater than zero, then another iteration is entrained. Otherwise, the program stops after another few steps.

The ability of computers to execute conditional-branch instructions—i.e., to modify the flow of control of their programs as a function of the outcome of tests on intermediate results of their own computations—is one of their most crucial properties, for every effective procedure can be reduced to a series of nothing but commands (i.e., statements of the form "do this" and "do that") interlaced with conditional-branch instructions. Moreover, only binary branching instructions (i.e., instructions of the form "if such-and-such is true, do this; otherwise do that") are needed. If the decision whether such-and-such is true or not itself involves complex procedures, these too can be cast into the framework of commands and binary (i.e., two-way) branching instructions.

I have now drawn a fairly accurate outline—albeit only an outline—of what a program is and of how a computer executes a program. In the course of the argument, we reduced a very wordy worksheet to a highly compact code, and then reduced that code still further, to a rigidly formatted bit string. We assumed all along that none of these reductions were accompanied by the loss of any essential information. We hope, in other words, that the string of 0's and 1's which is the final product of all our transformations means the same thing as our original worksheet. To test whether or not it does, we must know what each means separately.

The ability of computers to execute conditional-branch instructions is one of their most crucial properties.

What does the worksheet we have made up mean? To an illiterate it may mean an opportunity to build a paper airplane; and to a mother who believes her ten-year-old child composed it, that her child is a budding genius. But in our context, we see it as a set of instructions. Hence its meaning to us is the action someone who understands those instructions takes when he obeys them. The meaning of the computer code must then be the action a computer takes when it interprets that code as instructions and obeys them. We are confident we know what a person following the instructions written on the worksheet will do, because they are written in a language he and we share. What a computer will do, given the code we prepared, is entirely a function of its design. That design determines the meaning of any string of 0's and 1's given a computer. If the particular bit-string we have developed is to mean the same thing to a computer as what our worksheet means to a human calculator, then the computer must have components functionally equivalent to each of the operations we ask our human calculator to perform and to every resource we make available to him.

A computer code is the action a computer takes when it interprets that code as instructions and obeys them.

I have, of course, been very careful to structure this discourse about workers, supervisors, A boxes, P boxes, and so on, in such a way that these resources and functions would be analogous to components and functions of real computers. In fact, real computers do have A boxes, namely, registers called accumulators, and P boxes, namely, registers called program counters. Real computers do store their instructions in sequentially numbered registers, and they do manage the flow of control in roughly the way I have pictured. Modern computers are much more complex, both in their structure and in their functions, than my simple sketch of one kind of computer reveals, to be sure. But the fundamental operating principles I have described do provide a starting point for understanding even them.

When speaking of Turing machines, I asserted that a language is a game played with a certain alphabet and governed by a set of transformation rules, and that a computer is an embodiment of those rules. That assertion applies with equal force to modern digital computers. Procedures can, as we have seen, be encoded as bit strings whose format is dictated by certain design features of the computer for which they are intended. The primitive vocabulary that a programmer may employ is determined by the operations built into the computer. For example, a particular computer may have the square-root function built in as one of its primitive operations. The language corresponding to that computer would contain an operation code corresponding to that function, in effect, a verb meaning "take square root." In the language determined by some other computer, one that does not have "take square root" as a primitive operation, the square-root function can be realized only by a subroutine, in effect, a verb clause composed of more elementary terms. I wish to emphasize again that every computer determines a language, its machine language. The alphabet of the machine languages of all modern computers is the set consisting of the two symbols "0" and "1." But their vocabularies and their transformation rules differ widely.

Do computer programmers then work in the language of the machine they are instructing? I.e., do they actually compose complex procedures in terms of long bit strings? No. We have seen that it is straightforward and easy to translate the expression

(ADD 123)

into the bit string

00110101000000000000000001111011.

A computer is a superb symbol manipulator. It is relatively easy to design a procedure that will convert a program, such as the one in Table 2, into its corresponding bit string. Indeed, such procedures can also be made to handle mundane, but often far from trivially easy, bookkeeping tasks, such as the detailed assignments of storage locations. Programs that do this sort of work, i.e., that convert programs written in the kind of notation shown in Table 2 into machine language, are called *assemblers.* The language from which they convert, i.e., the language they translate, is called an *assembly language.*

A computer is a superb symbol manipulator.

Programs written in assembly language are, of course, much easier for people to read than programs in machine language. Since assemblers are themselves fairly complex programs, they too are written in assembly language, namely, the language they themselves translate. There is no paradox in this. An assembler is a procedural embodiment of well-determined rules that tell how to transform expressions composed on a certain alphabet into expressions composed on a much smaller alphabet. It therefore defines a language, its assembly language. Since it is designed to translate any legal assembly-language text into machine language, there can be no mystery about its ability to translate the text that constitutes its own definition as well.

I said earlier that a program transforms a computer into another computer. A general-purpose computer loaded with nothing but a square-root program, for example, has been transformed into a special-purpose computer capable of computing only square roots. An assembler, then, also transforms the computer into which it is loaded. The transformation it induces has important consequences: a programmer who instructs a computer using only an assembler need never learn the language determined by the computer itself, i.e., its machine language. In an important sense, he never sees the machine he is actually addressing; he sees and works with a symbolic artifact that, for him, *is* the machine.

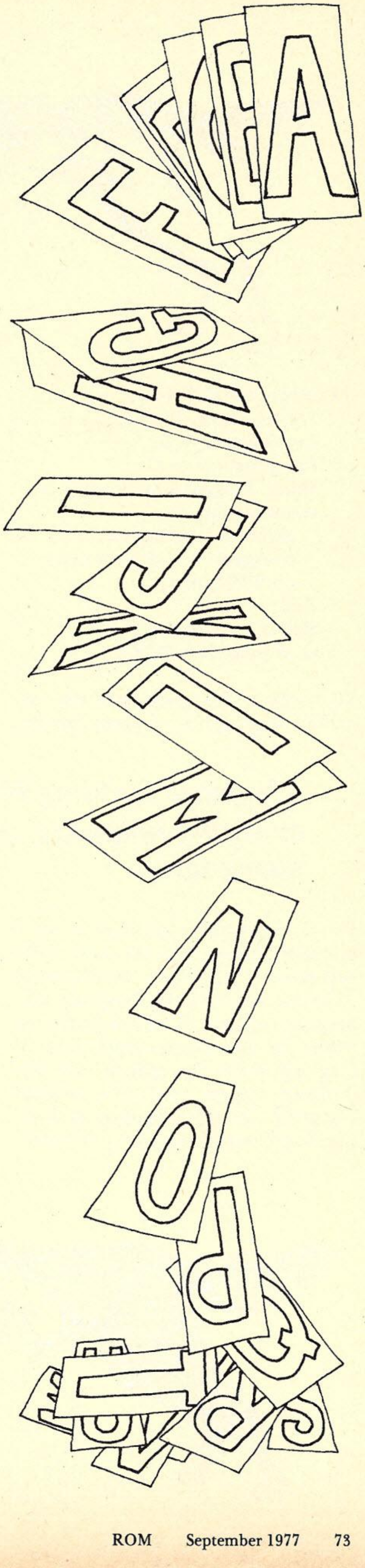
It did not take long for programmers to realize that the symbol-manipulating power of the computer could be employed to translate still "higher level" languages. Consider, for example, the following fragment of assembly language text:

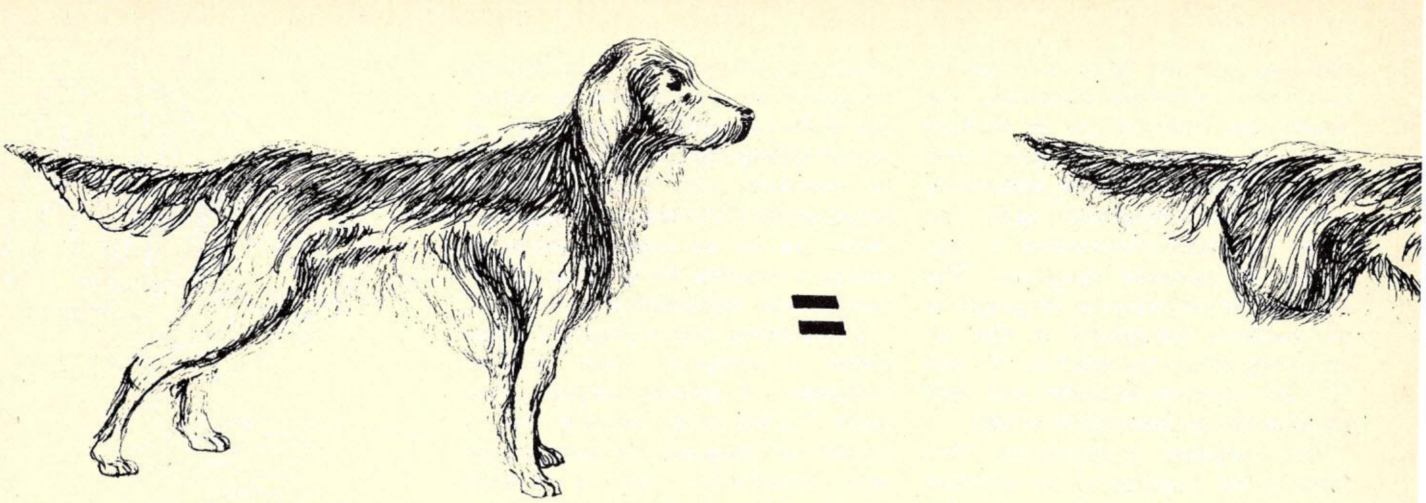
GET *a*.
ADD *b*.
STO *c*.

With even as little as we covered in the preceding pages, we can see that that fragment must be an encoding of the algebraic expression

$$c = a + b.$$

Clearly a very simple procedure will do to translate the latter expression into the assembly-language fragment shown. Now, without fretting much over details, it may be said correctly that the following is an encoding of the square-root procedure with which we have already had so much experience.





```
Procedure SQRT (n):
   Let tolerance = .001;
   Let oldguess = 0;
   Let newguess = 1;
   While | oldguess-newguess | >
   tolerance do:
      oldguess = newguess;
      newguess = (oldguess**2 +
      n)/2*oldguess;
   End;
   Result = newguess;
End of procedure SQRT.
```

The basic building blocks used to compose assembly-language programs are utterly elementary.

To translate that procedure into assembly language is relatively simple. We did essentially just that in some preceding pages. It is far more difficult however, to write a procedure to translate into assembly language any program written in the language of which the above is a sample text. It may appear at first glance that this difficulty results from the obvious complexity of the language as compared to the stark simplicity of assembly languages. It is true that the translator must be able to handle a very great variety of syntactic forms, many of which, especially as they occur in combinations, are the source of truly difficult technical problems. Still, these are not the hardest problems.

We must ask what it means to say that the procedure we have shown is a "sample text" of some language. Of what language? The problem of first priority, and by far the hardest one, is to design the "higher level" language at all. The formation and transformation rules of a language (as of a game) are, after all, as much a system of constraints, of prohibitions, as of permissions. The very fact that the basic building blocks used to compose assembly-language programs are so utterly elementary allows the programmer an enormous number of degrees of freedom. An assembly language for a particular machine is virtually a mnemonic transliteration of its machine language. The designer of an assembly language is therefore hardly confronted with questions of meaning. But the designer of a higher-level language must decide the kinds and number of degrees of freedom to be allowed to authors writing programs in that language. In a very real sense, every freedom an author exercises in making a choice is one he has usurped from the programmer who is to use his product. Consider a very simple example, namely, the expression

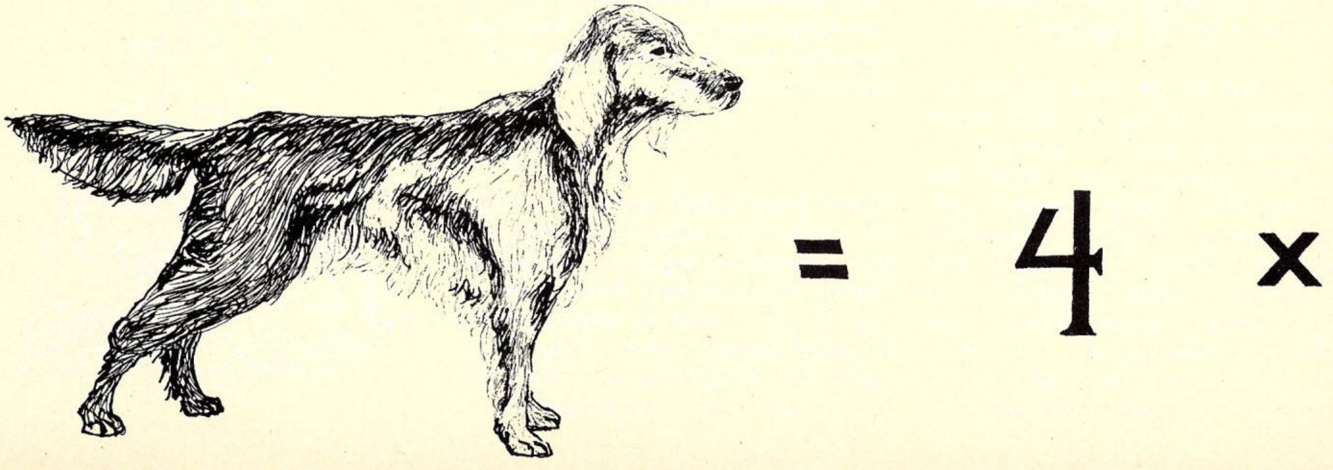
$$d = a + b * c$$

(where "*" is the multiplication operator). A programmer is free to write either the assembly code

```
GET a,
ADD b,
MPY c,
STO d,
```

which means that *c* is to be multiplied by the sum of *a* and *b* and the product stored in *d*, or he can write

```
GET b,
MPY c,
ADD a,
STO d,
```



which means that a is to be added to the product of b and c and the sum stored in d. But the higher-level language statement,

$$d = a + b * c,$$

can have only one interpretation. Which one it is to be is determined by the detailed design of the translator, hence by the designer of the language. Of course, this is not to say that the expressiveness of the higher-level language is necessarily limited. The language may, for example, permit the two different interpretations to be expressed by

$$d = (a + b) * c$$

and

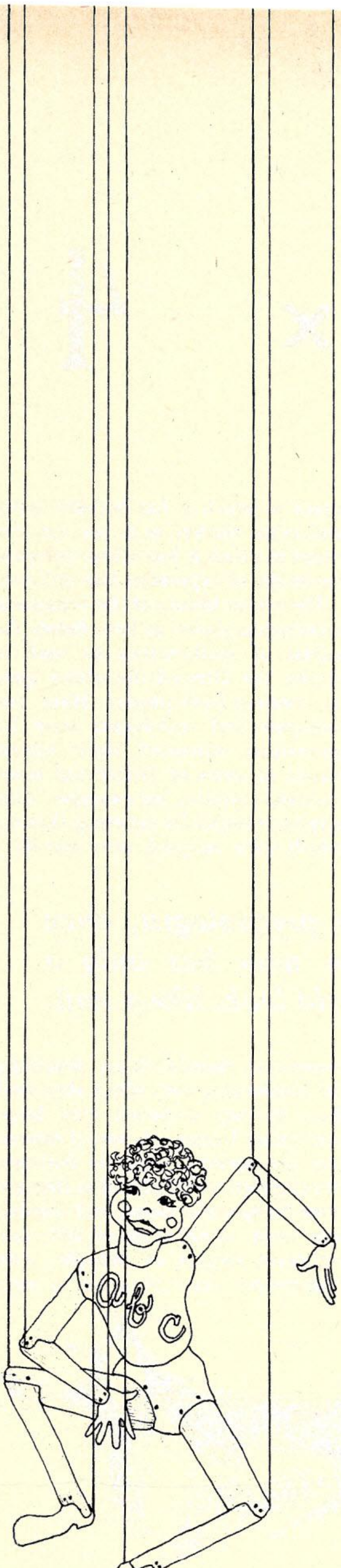
$$d = a + (b * c),$$

respectively. Although this extremely simple example barely touches it, the question is not what procedures are expressible in a higher-level language—most such languages are in fact universal in Turing's sense—but what programming style the language dictates. Abraham Maslow, the psychologist, once said that to a person who has only a hammer, the whole world looks like a nail. A language is also a tool, and it also, indeed, very strongly, shapes what the world looks like to its user.

A solitary programmer writing a program to solve some problem incidental to his private research need hardly concern himself over the way his program might structure the worldview of anyone besides himself. Higher-level languages, however, are intended to be very much public languages, in two important senses: they are intended to be used by very large numbers of programmers as the languages in which they instruct their machines; and they are also intended to be used as languages in which people are to communicate to one another the procedures they have composed. Indeed, the success of a particular higher-level programming language is measured largely by the extent to which it has become dominant in the market, so to say, i.e., the extent to which it has driven competitive modes of expression into oblivion.

Abraham Maslow, the psychologist, once said that to a person who has only a hammer, the whole world looks like a nail.

The recent history of the behavorial sciences has shown us how deeply the success of mathematics as used in physics has affected disciplines quite far removed from physics. Many psychologists and sociologists have for generations discussed their subject matter in terms of differential equations and statistics, for example. They may have begun by believing that the calculi they adopted were merely a convenient shorthand for describing the phenomena with which they deal. But, as they construct ever larger conceptual frameworks out of elementary components originally borrowed from foreign contexts, and as they give these frameworks names and manipulate them as elements of still more elaborate systems of thought, these frameworks cease to serve as mere



modes of description and become, like Maslow's hammer, determinants of their view of the world. The design of a public language, then, is a serious task, one pregnant with consequences and thus laden with extraordinarily heavy responsibility. I shall have more to say about that later.

For the moment I wish to emphasize that a higher-level programming language, such as that represented by the preceding fragment, is, in fact, a formal language. The meanings of expressions written in it are determined, of course, by its transformation rules, and these, in turn, are embodied in the procedures that translate it into assembly and ultimately into machine language. If, therefore, one had to say what a particular higher-language program means, one would have to point ultimately to its machine-language reduction and even to the machine corresponding to it. One would have to say "it means what that machine does with that code." However, that is not how such questions are answered in reality. For the translator is itself a program and therefore transforms the computer into which it is loaded into quite another computer, namely, a computer for which that language is its machine language. There is then an equivalence, not only between formal languages and abstract games, but between these and computing machines. To put it another way, there are important universes of discourse in which distinctions between languages and their machine embodiments disappear.

Lest this be thought a "merely" philosophical point of little practical consequence, I must say immediately that not only do most of today's programmers think of the languages they use as being their machines, very nearly literally so, but many, perhaps most, have no knowledge whatever of their computer's machine language or of the content and structure of the translators that mediate between them and their computers. This observation is not made as a criticism. After all, a higher-level formal language is an abstract machine. No one is to be faulted for using a language or a machine more congenial to his purposes than is some other machine, merely because the other machine is somehow more primitive. But the observation does raise an important question: If today's programmers are largely unaware of the detailed structures of the physical machines they are using, of their languages, and of the translators that manipulate their programs, then they must also be largely ignorant of many of the arguments I have made here, particularly of those arguments concerning the universality of computers and the nature of effective procedures. How then do these programmers come to sense the power of the computer?

Their conviction that, so to say, the computer can do anything—i.e., their correct intuition that the languages available to them are, in some nontrivial sense, universal—comes largely from their impression that they can program any procedure they thoroughly understand. That impression, in turn, is based on their experience of the power of subroutines and of the reducibility of complex decision processes to hierarchies of binary (i.e., two-way branching) choices.

A subroutine is, as I have said, a program whose input-output behavior we understand without necessarily understanding how it converts the input we give it to the output it delivers. A programmer faced with a difficult subproblem can always pretend there exists a subroutine that solves that subproblem. Perhaps he will find it in a program library somewhere. But even if he doesn't, he can attend to it after he has worked out the strategy for his main problem. If, for example, his main problem is to write a program to enable a computer to play chess, he will certainly need a subroutine that will produce a chess move, given a configuration of pieces on a chess board, and an indication of whose move it is, black or white. The input to such a subroutine must be some representation of the pieces on the chess board and, say, a "0" if it is white's move or "1" if it is black's. Its output is a chess configuration in the same representation as that employed in the input. Once he has decided on suitable representations, and that may be a nontrivial task in itself, he is free to "use" a move-generating subroutine in the further development of his main program as if it existed. If he is part of a programming team, he may well describe the subroutine's desired input-output behavior to a colleague who will then write it for him. In any case, he can go on with the strategic analysis and even the programming of his main problem without having to keep lower-level details constantly in mind.

I have already said that in computers and computation larger entities are built out of smaller ones. In a way, subroutines allow us to say the opposite as well. For with their aid we can break a very large and complex task into a set of smaller tasks, and each one of these into still smaller ones, and so on. Were that not so, no one would dare to undertake such monumental tasks as computer-controlled air-traffic management or the computer simulation of a large business. Of course, the idea of solving a problem by breaking it up into smaller problems and then solving them, and so on, is far from new. But, until the advent of the computer, huge hierarchical problem-solving systems always had to depend on people to carry out the designs of the master strategists. People, unlike computers, are not altogether reliable executors of even the most explicit instructions. The computer programmer's sense of power derives largely from his conviction that his instructions will be obeyed unconditionally and that, given his ability to build arbitrarily large program structures, there is no limit to at least the size of the problems he can solve.

> ***People, unlike computers, are not altogether reliable executors of even the most explicit instructions.***

I am finally in a position to comment on an erroneous impression undoubtedly left by something I wrote near the beginning of "Computer Power and Where It Comes From" (Vol. I, No. 2). There I wrote, "Machines do only what they are made to do—and that they do exactly." We have now seen how we can instruct computers to do things. If we take seriously, as we should, the statement that a program transforms one computer into another, then we have even seen how we can "make" a machine to do something. But the impression I have probably engendered is that programs must necessarily be in a form, perhaps thinly disguised, of a sequence of instructions that must be executed more or less sequentially.

I have already exhibited evidence to the contrary. The quintuples that govern the behavior of a classic Turing machine do not have to be in any specific order, nor are they obeyed sequentially. A typical such quintuple, cast in the form of a rule of a game, says:

> "If you are in state *u* and if the symbol you are currently scanning is *x*, then do. . . ."

The Turing machine literally searches its tape for the rule applicable to its situation. The order in which the rules are written is totally irrelevant to the machine's ultimate input-output behavior.

There are two important points to be made in this connection. The first has to do with the idea of searching for a rule. The Turing machine searches linearly for a rule to apply. The information on its tape is, after all, in linear order. But in modern computers other searching regimens are possible. In particular, one can write programs whose function is to compute search rules. Then, just as a programmer may know a procedure for computing square roots of numbers but not know the square root of any particular number, let alone of all the numbers that may be given to his procedure, so he may know how to specify the construction of search rules but not know any of the rules his specifications may produce. A computer that obeys rules it finds by carrying out such computed searching regimens does indeed do only what it has been made to do. But the "making" is at least one, possibly many more, removes from the original maker, i.e., from the programmer. I shall discuss some implications of this again.

The second point has to do with how rules weakly analogous to Turing-machine quintuples may be given to computers. Certainly rules may be entered in the form of programs of the kind I have exhibited. But let us leave computers for a moment and consider how a person would wrestle with the question, "What's the area of the dish?" First he would assume that the dish he is being asked about is the last dish mentioned. He would therefore search his memory for the last mention of a dish. If that mention was, "The area of the dish is such-and-such," he would be able to answer quickly. Otherwise he would search his memory for statements of the form, "The area of so-and-so is. . . ." i.e., for some stored knowledge of how one answers questions about the area of things. Suppose he finds the statement, "The area of a circle is pi times the radius of the circle squared." He might think, "If the dish has a radius, it may be a circle and then I could apply this rule." If it turned out that the last mention of a dish was "The radius of a dish is 10," the rest would be obvious. Below is the typescript of a "conversation" with a computer conducted via an electric-typewriter computer console (the lines printed all in capitals are the computer's responses):

```
The area of a circle is pi times the
   radius of the circle squared.
The area of a sphere is pi times
   the diameter of the sphere
   squared.
The area of a square is the side of
   the square squared.
The diameter of an object is twice
   the radius of the object.

The radius of a ball is 10.
What is the area of the ball?

THE ASSUMPTION THAT
   BALL IS CIRCLE IS CON-
   SISTENT; SHALL I GO ON?

No.

THE ASSUMPTION THAT
   BALL IS SPHERE AND OB-
   JECT IS CONSISTENT;
   SHALL I GO ON?

Yes.

IT'S 1256.6359.

What is the diameter of the ball?

IT'S 20.

The radius of a dish is 10.
What is the area of the dish?

THE ASSUMPTION THAT
   DISH IS CIRCLE IS CON-
   SISTENT; SHALL I GO ON?

Yes.
```

```
IT'S 314.15899.

The radius of a globe is 10.
A globe is a sphere. A sphere is an
   object.
What is the area of the globe?

IT'S 1256.6359.

The side of a plate is 10.
What is the area of the plate?

THE ASSUMPTION THAT
   PLATE IS SQUARE IS CON-
   SISTENT; SHALL I GO ON?

Yes.

IT'S 100.

When I say board I mean plate.
What is 3 times the side of the
   board?

IT'S 30.

Please call that x.
Let f(z) be z squared minus z.
Compute f(x).

IT'S 870.
```

The program that made a computer do this is best seen, not as a set of instructions of the form "If this, do that," but as a collection of rules of evidence. It uses these rules to decide how assertions it may be given are relevant to questions it may be asked. The computer's behavior is therefore governed as much by the assertions it is fed, e.g., "A globe is a sphere," as by its instructions. The oft-repeated truism that a computer can do only what it is told to do thus turns out, like most truisms about complex matters, to be, to say the least, problematical. There are many ways to "tell" a computer something.

The idea that a person can write a program that embodies anything he "thoroughly understands" is at least equally problematical. Understanding something always means understanding it at a certain level. An actuary uses some fairly sophisticated mathematical tools whose fundamentals he almost certainly doesn't understand or care anything about. Everyone who makes change uses arithmetic, but very few people know or care much about the beautiful axiom system on which arithmetic is based. In effect, we all constantly use subroutines whose input-output behavior we believe we know, but whose details we need not and rarely do think about. To understand something sufficiently well to be able to program it for a computer does not mean to understand it to its ultimate depth. There can be no such ultimate understanding in practical affairs. Programming is rather a test of understanding. In this respect it is like writing; often when we think we understand something and attempt to write about it, our very act of composition reveals our lack of understanding even to ourselves. Our pen writes the word "because" and suddenly stops. We thought we understood the "why" of something, but discover that we don't. We begin a sentence with "obviously," and then see that what we meant to write is not obvious at all. Sometimes we connect two clauses with the word "therefore," only to then see that our chain of reasoning is defective. Programming is like that. It is, after all, writing too. But in ordinary writing we sometimes obscure our lack of understanding, our failures in logic, by unwittingly appealing to the immense flexibility of natural language and to its inherent ambiguity. The very eloquence that natural language permits sometimes illuminates our words and seems (falsely, to be sure) to illuminate our undeserving logic just as brightly. An interpreter of programming-language texts, a computer, is immune to the seductive influence of mere eloquence. And words like "obviously" are not represented in the primitive vocabularies of any computers. A computer is a merciless critic. Therefore the assertion that one understands a thing sufficiently well to be able to program it is, first of all, an assertion that one understands it in very particular terms. In any case, it can be no more than a boast that may well be falsified by experience.

A computer is a merciless critic.

A computer is immune to the seductive influence of mere eloquence.

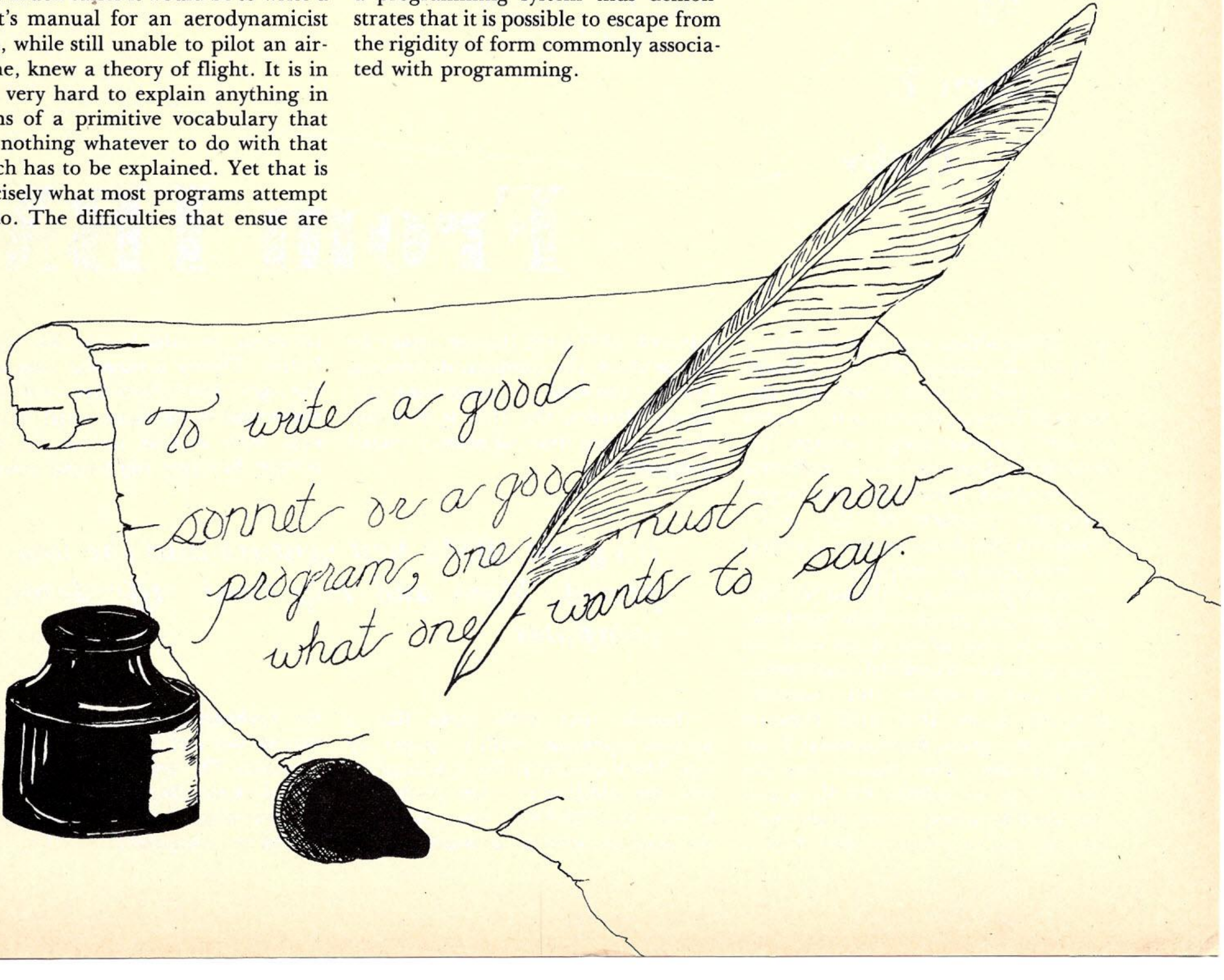
The other side of the coin is the belief that one cannot program anything unless one thoroughly understands it. This misses the truth that programming is, again like any form of writing, more often than not experimental. One programs, just as one writes, not because one understands, but in order to come to understand. Programming is an act of design. To write a program is to legislate the laws for a world one first has to create in imagination. Only very rarely does any designer, be he an architect, a novelist, or whatever, have so coherent a picture of the world emergent in his imagination that he can compose its laws without criticism from that world itself. That is precisely what the computer may provide.

One must, of course, learn how to use criticism. In many cases the computer's criticism is very sharp and cannot be ignored; a program doesn't work at all, or delivers obviously wrong results. Then there is no question that something has to be redesigned. Computers are maddeningly efficient at stumbling over purely technical, i.e., linguistic, programming errors, but stumbling in a way that disguises the real locus of the trouble, e.g., just which parenthesis was misplaced. Indeed, many professional programmers believe that their craft is difficult because the languages with which they must deal have rigid syntactical rules. There is therefore a persistent cry for natural-language, e.g., English, programming systems. Programmers who hold to this belief have probably never tackled a truly difficult problem, and have therefore never felt the need for really deep criticism from the computer. It is true that in order to write one has first to master the syntactic rules of one's chosen language. But then one must also have thought through what one has to say. Literary criticism is not the business of calling attention to spelling errors and to technical violations of grammatical rules. It has to do with substantive ideas. The literary critic has to know much. A real reason that programming is very hard is that, in most instances, the computer knows nothing of those aspects of the real world that its program is intended to deal with. It is in a position analogous to, say, that of a teacher of English grammer who has been given an airplane pilot's manual and left at a plane's controls in midair. He may be a very skilled grammatical faultfinder, but as to how to fly an airplane, he knows nothing. The manual had therefore better explain everything in terms sufficiently primitive that even an earthbound schoolteacher can understand them. How much easier it would be to write a pilot's manual for an aerodynamicist who, while still unable to pilot an airplane, knew a theory of flight. It is in fact very hard to explain anything in terms of a primitive vocabulary that has nothing whatever to do with that which has to be explained. Yet that is precisely what most programs attempt to do. The difficulties that ensue are no more rooted in syntactic rigidities than is, say, the difficulty of writing a good sonnet rooted in the rigid form demanded by that class of poem. To write a good sonnet or a good program, one must know what one wants to say. And it helps enormously if one's critic shares one's relevant knowledge base.

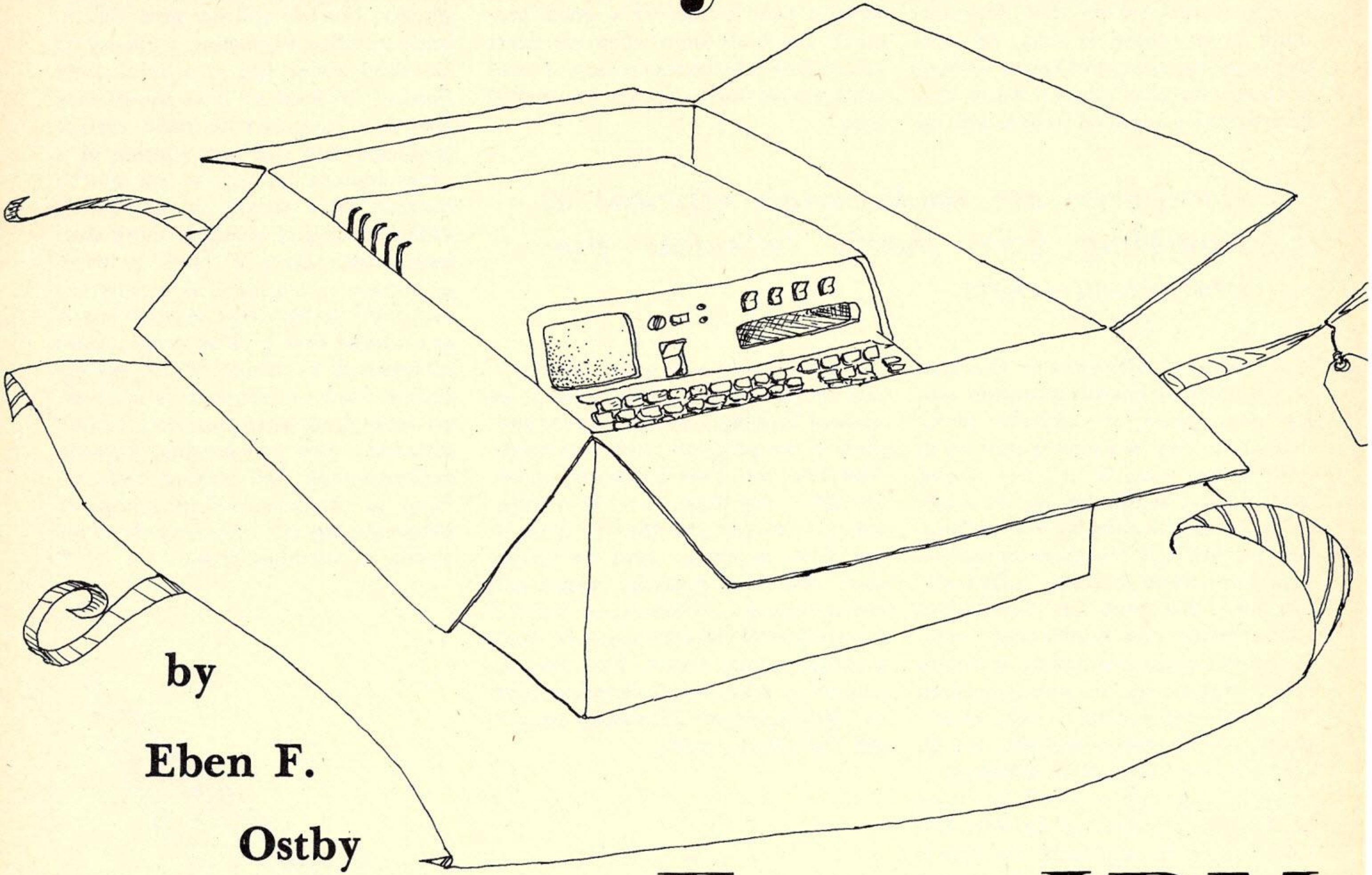
Computers are maddeningly efficient at stumbling over purely technical programming errors.

We have seen a program that consists in part of assertions given it in natural language. These may be supplied to the program at various times. The whole set of assertions may therefore grow very large. It is, in a sense, easier to program in this style than in the more orthodox ones commonly used. And it is possible, using such programming techniques, to build a knowledge base into a computer. Such a programming system thus demonstrates that it is possible to escape from the rigidity of form commonly associated with programming.

But we mustn't make too much of this. A lazy student's list of useful physics formulas may enable him to solve many schoolbook problems in physics, but will still not give him an understanding of physics, a theory he can think about. Just so, a set of assertions of the kind we have shown may enable a computer to solve certain problems. But this says nothing of a programmer's or of a computer's mastery of a theory or of either's understanding of anything more than how to use a group of "facts" to arrive at certain conclusions. A computer's successful performance is often taken as evidence that it or its programmer understand a theory of its performance. Such an inference is unnecessary and, more often than not, is quite mistaken. The relationship between understanding and writing thus remains as problematical for computer programming as it has always been for writing in any other form. ▼



Personally Yours From IBM



by

Eben F.

Ostby

Its introduction was preceded by a flurry of rumors. We were advised of its arrival by none other than Dr. Kenneth Iverson, the man who created the APL programming language. Behind its development were a small crew of some of the finest men in the largest computer company in the world. What was this dramatic new product? It was a personal computer.

You might well ask why all the fanfare over just another little machine, yet another one of the many desk-top microprocessor-based little computers. The reason is simple: this particular computer is the IBM 5100 Personal Computer—giant International Business Machines' first venture into the price range affordable by Mom and Pop sized businesses—even some individuals. And everything IBM does is watched with great interest, since by its size alone, the company determines much of the future of computing machines. Besides, the 5100 is in many ways different from all other personal computers.

It's giant IBM's first venture into the low-priced, Mom and Pop sized, affordable computer.

Outside, the 5100 looks like a bloated typewriter, with no paper up top. The keyboard is like a Selectric's, with the addition of a key at the left labeled COMMAND. Also, many of the keys have symbols or words printed on them, in addition to the regular letters. There's a numeric keypad on the right, which looks like a calculator embedded in the front panel. Finally, keys with various arrows on them occupy the upper right-hand corner of the keyboard. Instead of a printing mechanism and paper, there's a little five-inch TV screen above the keyboard. A slot for special cassette tape and an on-off switch complete the layout of the computer.

Illustrated by Eben F. Ostby

From this description, the 5100 could be any one of a number of personal computers, perhaps with a few frills. But the operation of the 5100 is unlike other small computers, since it was designed to keep the machine-level workings of the computer completely transparent for the user.

You have all the power of a huge 360 at your fingertips—except there are a few hitches.

With the 5100, you don't load a BASIC compiler into memory, you just program the computer in BASIC. Or if you prefer, you can program it in APL. It acts as if all it understood were higher-level languages, as if it had no machine language of its own. It's sort of like having a car with an automatic transmission: you don't have to mess with the mechanics of shifting (or machine language programming). In this way, too, the costs of software are included in the cost of the computer—in fact, you can't buy a language compiler for the 5100, you can only get what comes with it: APL or BASIC.

The chief ingredient of the 5100 is ingenuity.

It's worth noting that a few other personal computers come with software built in, but none has the choice of either APL, or BASIC, or both, that the 5100 has.

What's it like to use the 5100? It's much like using a terminal connected to a time-sharing computer service. Turn it on, and switch it to APL or BASIC, as you prefer. Before you get a chance to do anything, the computer runs through a series of tests designed to check out all of its inner workings. When the words CLEAR WS appear on the screen (if you're using BASIC, READY will appear), the computer has tested itself out and is ready for you to program it. You can either write programs or you can use the computer in a desk-calculator mode. (You enter the numbers; it displays the answer.) Whatever you type, along with the computer's response, appears on the TV screen in small letters. When you're done, you can save what you've calculated on the cassette tape: it will hold 204,800 characters of information, or 51,200 integers. (Grabbing the nearest book at hand, I find that that's just about enough to store every telephone number on Cape Cod.)

The 5100 seems impressively fast. Using the APL function that sorts number lists, you can sort a thousand numbers in order in seconds. Additions, multiplications, even complex operations like taking logs are all executed instantaneously. It really appears that you have the power of a huge 360 at your fingertips. And you do, except there are a few hitches.

Because of the way it was designed, some operations on the 5100 are very slow. For instance, a simple expression that removes duplicate letters from a sentence gets so slow when used on long sentences, that a thousand-word sentence takes almost a minute to process. Even if you're a patient person, it's likely that you'd get bored waiting for that—especially after the computer just got finished doing two thousand additions in a few seconds. Other operations, involving "bit strings" (numbers that can only take on the values 1 or 0) take even longer, because the machine insists on using an involved process to extract the numbers from their encoded state. Also, the speed at which programs are executed is just not that fast, so that it's sometimes necessary to program in what appears to be a roundabout way to get the machine to run at a good clip.

The 5100 has a respectable-sized memory. Because the language processors are stored in a separate memory (called executable read-only storage), almost all of the "normal" memory (read-write storage) is used for storing user programs and data.

ROMtutorial ROMtutorial

Microprocessor: The "thinking" section of a computer is called the central processing unit (CPU) or just the processor. If it's so small that you need a microscope to examine it, it's called a microprocessor.

Keypad: A small section of a keyboard, like the buttons on a calculator.

BASIC: A computer language which allows anyone to use the computer and get instantaneous feedback as to whether they are doing OK or making a mistake.

APL: A concise, powerful, programming language. (See "APLomañia" in August *ROM*)

Higher-level language: Programming language that is further removed from the machine's internal workings. Usually, a higher-level language will be easier to program in than a lower-level language.

Machine language: The internal programming language that the computer uses. Usually, different computers have different machine languages. These may or may not be compatible.

Software: The programs that are run on a computer. A computer system consists of hardware—the computer and its accessories—and software—the programs which make the hardware work the way you want it to.

Desk-calculator mode: When a system is in desk-calculator mode, it acts like a calculator: numbers and instructions are entered, and the computer responds with the results of the calculations desired.

ROMtutorial ROMtutorial

BASIC compiler: A program which converts a program written in BASIC into the computer's machine language.

Executable read-only storage: A type of memory whose contents cannot be modified by the computer. A program can be stored and executed (run) in this type of memory.

K: From kilo, which is in turn from the Greek *khilioi,* a thousand. Except in computerese it usually means 1024 (2^{10}), not 1000 as in metric measurement.

Read-write storage: Memory, like executable read-only-storage, which *can* be modified under the computer's control.

Byte: A piece of information consisting of eight bits. It has 256 possible values.

Floppy disk: An information storage device which uses a "disk" which looks like a 45 rpm disk but is coated with a magnetic surface like that used on a recording tape. The disk is flexible, or "floppy."

Interface: The boundary between different sections of a computer, or between the computer and the outside world. The circuitry which crosses the boundaries is also called the interface.

The smallest 5100 has 16K bytes of read-write storage, which is large by microcomputer standards, in addition to the read-only storage (ROS), which may be several times the size of the read-write storage. The largest machine has 64K bytes of read-write storage, which is pretty big by any standard. It's fortunate that the 5100 comes with lots of storage though, since one of the languages it uses, APL, achieves its great speed and power through the use of lots of memory. In fact, it wouldn't hurt to have even more memory for doing some complex operations in APL—but that's for the future.

One of these tapes can hold the whole heart of someone's business—it had better not break.

Since the 5100, like most personal computers, loses whatever it had in memory when it is shut off, a way of storing data and programs for use on another day is a necessity. The 5100 uses its cassette tape drive and tapes for this. The tapes are about twice the size of the cassettes you use in your stereo—bigger even than an eight-track tape—and they're far more elaborate. Built on a heavy (eighth-inch thick) metal base, the tapes contain a special drive system, so they feed smoothly. A mirror is used for determining when the tape is used up, and a little trap door opens only when the tape is inserted in the tape drive. They cost a bit more than regular cassettes, too: $20 apiece (minimum order: five tapes). But as an engineer pointed out to me, one of these tapes can contain the complete heart of someone's business—it had better not break. Even at $20 apiece they are not much more expensive than quality paper needed to type out an equivalent amount of information.

Emulation is used by manufacturers all the time to test out a new computer before it's even been built.

Reliable as the tapes are, they are primitive by the standards of big computers. The biggest problem is that they're rigid—meaning not that they're brittle, but that they have an inflexible format—and hence very slow. Tape is a sequential medium: when you write something on a tape, the next thing you write usually gets written right after the last thing. When you read data off a tape, you can only read the pieces of data in sequence, the sequence in which they were written. This is true for all tape systems, big or small. But on most big computers, you can make reading and writing easier, and avoid some of the problems of sequential access, either by reading and writing backwards as well as forwards, or simply by reading and writing arbitrary small parts of a tape. With the 5100 neither of these are possible. If you want to find the twelfth record on a tape, you have to read the first eleven first. If you want to replace the seventy-first record of a tape that has one hundred records, you have to read in the first seventy, write them out onto another tape, write out the revised seventy-first record, and read in and write out the remaining twenty-nine records onto the new tape. Not only does it take an incredible amount of time (sometimes over an hour), but you have to buy a second tape drive from IBM. Pretty clever.

If you want to make the 5100 work a bit more quickly, it is possible to buy (not from IBM—they don't make them) a floppy disk drive that will attach to the computer through an interface. This device lets you make all the updates and random record accesses you wanted to do with the tape drive, but couldn't quite manage.

The most important other "extra" you can buy for a 5100 is the optional printer. The little TV screen is okay—it doesn't waste paper—but the sixteenth-inch-wide characters disappear permanently when they scroll up beyond the limits of the screen. As new lines of type appear on the screen, the old ones move up, and the top lines are "pushed off" the screen.

For reports, records, or any time permanent output is needed, the printer is a must. It operates at 80 or 120 characters per second. That's pretty fast: the faster model can fill a large sheet of computer paper with printing in just a minute. The slower model will fill a sheet of typing paper with printing in under a minute. The printers themselves can go this fast—but the amount of time it takes the computer to prepare a page full of characters may be quite a bit longer.

It's no matter that this discrepancy exists though, since the computer stops processing every time the printer prints. This means that every time you print a page of text on the 5100's printer, the computer is wasting sixty seconds just waiting around for the printer to finish. Those sixty seconds could come in handy some day.

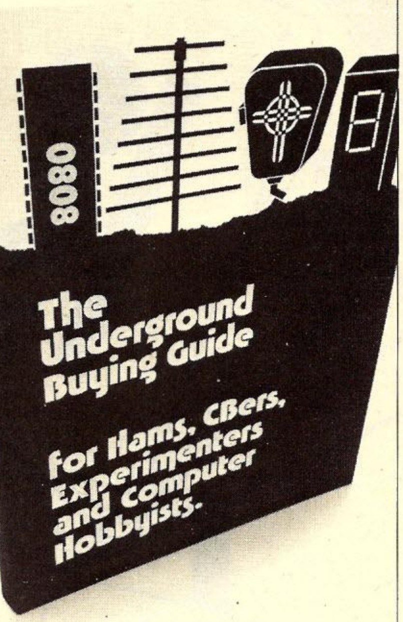
> *The 5100 is wasting sixty seconds just waiting for the printer to turn out a page. Those sixty seconds could come in handy some day.*

All of the components of the 5100—the computer itself, the printer, the tape drives—were put together in record time by a bunch of dedicated engineers. Rumor has it that the head of the 5100 project was *so* dedicated that he voluntarily moved from sunny California to snowy Minnesota in the dead of winter, just to work on the design of the computer. Whether or not this is true, someone in the project must have been inspired: this complex computer system was pieced together from scratch in around fifteen months. The chief ingredient of the 5100 is ingenuity: almost all of the components of the system were "stock" units—it just took some clever work to get the pieces together in a satisfactory way. The keyboard is taken, more or less directly, from standard keyboards that IBM uses in other equipment. The electronics weren't designed exclusively for the 5100, either. The printer is one that has been in use for some time in new IBM terminals. Even the most expensive and hard-to-adapt component of any computer system—the software—is a standard IBM system.

The APL interpreter, for example, is by and large the same system used on the huge IBM System 360. In order to run it, the 5100 emulates a 360: it simulates the operation of the larger computer. This, actually, is not a new idea; in fact, emulation is used by computer manufacturers all the time to test out a new computer before even a sample computer has been made. Using the same technique, software can be written for a new computer so that it will be ready when the production models are ready, too, and emulation is used frequently by businesses that are converting to a newer, better computer. But the 5100 emulates more or less exclusively—and as a result, the APL software for the 5100 was developed in a fraction of the time it otherwise would have taken. Of course, the user pays the price of such quick development—the 5100 spends lots of its time (maybe half) just emulating the 360. Thus your programs take that much longer to run.

Still, despite its drawbacks, the 5100 is a uniquely powerful computer. Since it was introduced less than two years ago, it has gained users all over the globe. It is particularly popular with small businesses, because it is flexible and easy to program. A camera shop in western Connecticut had a couple of college kids program it to handle the books. A little company in Westchester is using it to keep track of warranty claims and inventory. A small savings bank uses it to handle its ledger. Doctors and dentists use it for billing, preparing reports for insurance claims, sending information on patients for testing, and a host of related uses. A pharmaceutical firm is using it for analysis of data on new drugs. CPAs use it for accounting. Universities use it to drive graphics displays. Already, a host of consulting firms have sprung up, specializing in programming the 5100.

The 5100 has, in short, created a niche for itself. Maybe it wouldn't have, had it not been backed by the giant of the industry. But, like it or not, the giant of the industry has created a giant among tiny computers. ▼


8080
The Underground Buying Guide
for Hams, CBers, Experimenters and Computer Hobbyists.

Where to get it.

Equipment, parts, supplies and services. Hard to find and standard items at bargain prices.

Over 600 places to find transceivers, antennas, surplus, new and used equipment, μPs/computers, ICs, components, assortments, assemblies, discounted items, test equipment, peripherals, etc. Hundreds of large and small mail order sources.

A complete directory divided by sources, items and locations. Saves countless hours of shopping. Easily pays for itself through comparative buying. Contains no advertising.

Rush my order. I enclose $5.95 plus 55¢ postage and handling. Californians add 39¢ sales tax. Full refund if not completely satisfied within 10 days.

Name ____________

Address ____________

City/State/Zip ____________

Primary interest: Amateur Radio ☐ CB ☐ Experimenting ☐ μPs/Computers ☐

Send to: Peninsula Marketing
Dept. P
12625 Lido Way
Saratoga, CA 95070

Deal yourself in™ on the biggest personal computing show of the year!

FREE! SEMINARS, FORUMS, TECHNICAL TALKS

ENIAC by Dr. John Mauchly, the co-inventor of ENIAC
PROGRAMMING ENIAC by Mrs. John Mauchly
SAM 76 by Claude Kagan of Western Electric Co., an interactive symbol system manipulations system which grows with the user.
TELECOMMUNICATIONS FROM THE TERMINAL USER'S VIEWPOINT by David L. Peters of Vadic Corp.
INTRODUCING THE HEATHKIT COMPUTER PRODUCTS by Lou Frenzel of Heath Company
HOW MICROPROCESSORS ARE DESIGNED by Will Mathys of MOS Technology
THE FUTURE OF MICROS IN MEDICINE by Dick Moberg, Dept. of Neurosurgery, Jefferson Medical College, Philadelphia
THE HUMAN FACTOR by Andrew Singer of ROM Magazine
SHOULD MICROS BE USED FOR BUSINESS APPLICATIONS? by Frank J. Ponzio, Jr., of Mini Computer Suppliers, Inc.
ROBOTS by Tod Loofbourrow, author for Interface Age Magazine
GETTING INTO THE MICRO COMPUTER BUSINESS by Robert S. Jones, publisher of Interface Age Magazine
MUSIC FOR THE HOBBYIST, HARDWARE AND SOFTWARE by Malcolm Wright of Solid State Music
HAM RADIO APPLICATIONS by Dr. Robert Suding of the Digital Group
HANDICAPPED SYMPOSIUM by Dr. Robert Suding of the Digital Group
HOME MANAGEMENT SYSTEMS by Dr. Robert Suding of the Digital Group
FLOPPY DISK by Herbert G. Waite of PerSci Inc.
INTERFACING A HOME SELECTRIC by Charles Yates of Amateur Computer Group of New Jersey
MOVING UP TO AMATEUR RADIO by Chod Harris of the American Radio Relay League
OPERATING THROUGH AMATEUR SATELLITES OSCAR 6 AND 7 by Gary Tater W3HUC of AMSAT
THE PHASE III MICROPROCESSOR CONTROLLED AMATEUR SATELLITE by Tom Clarke WA3LND and Jan King W3GEY of AMSAT
MICROPROCESSOR APPLICATIONS FOR RADIO AMATEURS by Kasser G3ZCZ of AMSAT
WHAT PEOPLE ARE NOT GOING TO DO WITH HOBBY COMPUTERS by Stephen Gray of Creative Computing
APPLICATIONS OF MICROCOMPUTERS: THE MYTH AND THE REALITY by David Ahl of Creative Computing
INTRODUCTION TO COMPUTERS THROUGH THE BASIC LANGUAGE by Eri Golembo of Computer Mart of New Jersey
DYNAMIC DEBUGGING SYSTEM FOR THE 8080 CODE by Larry Stein and David Benevy of Computer Mart of New Jersey
MICROPROCESSORS FOR THE HOBBY MARKET TODAY AND TOMORROW by Dr. Adam Osborne of Osborne and Associates
GETTING STARTED WITH MICROCOMPUTER SOFTWARE by Dr. Christopher A. Titus, author of the Bugbooks
COMPUTERS AND MUSIC by Carl Helmers of BYTE Magazine

SATURDAY NIGHT BANQUET WITH OUTSTANDING SPEAKERS. Tickets are limited, first come, first served.

DOOR PRIZES

Thousands of dollars worth. You may be a lucky winner!

Session of the Proposed National Organization of Computer Clubs chaired by Sol Libes of the Amateur Computer Group of New Jersey and Richard Kusmack of Chesapeake Microcomputer Club.

COME TO PC '77 ...
Atlantic City, New Jersey
August 27 and 28, 1977

PC '77 offers you the most complete show of its kind ever held. Proven in '76 and acclaimed in '77 by all the major professional publications as the coming event of the year, this show is a 'must'. Make plans now to attend. Here are some of the scheduled events:

PRE-CONVENTION PROFESSIONAL SEMINARS

August 22-26 Technical Design Labs and Trenton State College Z80 Seminars at nearby Trenton State College.
Five software and four hardware seminars.

August 25, 26, 28 SYBEX Seminars at the Shelburne Hotel. Three intensive seminars: Introduction to Microprocessors, Programming Microprocessors, Microprocessors Applications.

August 24, 25, 26 TYCHON INC. Microcomputer Interfacing Workshop at the Shelburne Hotel.

August 26, 27 Osborne & Associates Microprocessors — Where they came from and where they are going, an analysis of all products on the market today. At the Shelburne Hotel.

MORE NEW PRODUCTS THAN EVER!

All the products you've been reading about in the ads will be on display at PC '77. Many companies will be showing exciting new products. HEATH COMPANY will display exclusively, for the first time, their complete computer line. SOLID STATE MUSIC, POLYMORPHIC SYSTEMS, THE DIGITAL GROUP, THOMAS INSTRUMENTATION, MOS TECHNOLOGY, TECHNICAL DESIGN LABS, SOUTHWEST TECHNICAL PRODUCTS, CROMEMCO, E & L INSTRUMENTS, THE INTERPRING GROUP, KENT-MOORE INSTRUMENTS, PERSCI INC, GEORGE RISK INDUSTRIES, MID WEST SCIENTIFIC, OSBORNE AND ASSOCIATES, EXPANDOR, QUAY CORP, MATRIX PUBLISHERS, CAMELOT PUBLISHING CO, HAYDEN BOOK CO, GAW ELECTRONICS, ENCLOSURE DYNAMICS AND SOROC TERMINALS will all be showing new products. Plan to attend!

OUTSTANDING COMPUTER HOBBYIST OF THE YEAR AWARD

This is an annual award presented to a person who has given outstanding service to others in the personal computing field with no commercial motives. Nominations are currently being accepted from individuals and clubs.

Personal Computing College

In-depth information through seminars, lectures and workshop sessions covering the personal computer field.

OSBORNE AND ASSOCIATES, INC.

August 26 & 27

"MICROPROCESSORS — WHERE THEY CAME FROM AND WHERE THEY ARE GOING. AN ANALYSIS OF ALL PRODUCTS ON THE MARKET TODAY." Dr. Adam Osborne will present a 6 hour seminar covering topics such as; Bringing order out of chaos; All microprocessors are not equal — each serves one market better than the other; Identifying those markets best suited to each microprocessor; Real sales volume anticipated for 1977; Comparisons including; 8085 vs. Z80, 8048 vs. F8, Etc. For information and reservations contact: Osborne and Associates Inc., Dept. PC77, P O Box 2036, Berkeley, CA 94702 (415) 548-2805.

TYCHON, INCORPORATED

August 24, 25 & 26

3 DAY MICROCOMPUTER INTERFACING WORKSHOP

A hands on experience for the participants where they will spend almost 50% of their time working on well documented Interfacing and Software experiments. Students deal with the microcomputer at the bus level, interfacing the computer using solderless breadboarding techniques and assembly language programs. Presented by Jonathan A. Titus and Dr. Christopher A. Titus, Authors of the famous "BUGBOOKS." For information and reservations contact: Tychon, Inc., c/o Shortness-Rawson, Dept. PC77, P O Box 2203, South Hackensack, NJ 07606.

SYBEX INCORPORATED

Thursday, August 25 & Sunday, August 28

INTRODUCTION TO MICROPROCESSORS

This intensive seminar is intended for all non-specialists who wish to acquire a broad understanding of the basic concepts and advantages of microprocessors. It explains how microprocessors work and it stresses methods, costs, advantages and disadvantages for the most important application areas of each type of microprocessor. What is needed to implement a system; how to use it; the impact on microprocessor-based systems; their evolution. Topics covered include: BASIC DEFINITIONS, SYSTEM COMPONENTS, MICROPROCESSOR APPLICATIONS, WHAT TO LOOK FOR, and IMPACT AND EVALUATION.

PROGRAMMING MICROPROCESSORS

This seminar describes in detail the internal operation of a microprocessor system including how instructions are fetched and executed, how programs are written and executed in typical cases (arithmetic and input-output). The goal of this course is to provide an overall understanding of the basic concepts of microprocessor programming. Requires an understanding of the main concepts in the INTRODUCTION TO MICROPROCESSORS course. It is recommended that these two seminars be taken together.

Friday, August 26

MICROPROCESSOR APPLICATIONS

This seminar presents in detail the main application techniques of microprocessors. Topics covered include INTRODUCTION TO MICROPROCESSOR SYSTEMS, APPLICATION TECHNIQUES, CASE STUDIES (industrial applications, medical and business applications, microprocessors in the home, and others), and EVOLUTION. For information and reservations contact: Sybex Inc., 2161 Shattuck Ave., Berkeley, CA 94704 (415) 848-8233.

TECHNICAL DESIGN LABS AND TRENTON STATE COLLEGI

August 22 through 26

PROGRAMMING IN BASIC FOR THE uC OWNER.
An introduction to the Basic language. From beginning to writing application
grams. Emphasis on TDL's 8K and 12K Basic for the Z80. Includes intro to T
FORTRAN.

ASSEMBLY LANGUAGE PROGRAMMING FOR THE Z80/8080
An introduction to assembly language programming. First covers 8080 instruction
then branches into the extra instructions available for the Z-80. Includes int
TDL's Z-80 Monitor and Macro-Assembler and also covers applications in interf
and control.

ADVANCED ASSEMBLY LANGUAGE PROGRAMMING FOR THE Z-80.
For someone who knows the 8080. Covers added Z-80 instructions and how to g
most out of them. Emphasis on use of TDL's Macro-Assembler. Application to c
logic replacement and process control.

FORTRAN IV
An introduction to the FORTRAN language. From beginning to writing of applic
programs. Emphasis on TDL's ANSI standard FORTRAN IV for the Z-80.

WORD PROCESSING WITH A TEXT EDITOR AND FORMATTER
Introduction to the use of the Text Editor and Text Output Processor for achi
basic word processing capabilities. Emphasis on TDL's Text Editors and Text O
Processors. Familiarization with system requirements.

INTRODUCTION TO HOBBY COMPUTING
A survey course dealing in an exploration of the Hobby computing field and de
the basics needed to be understood in order to get started.

DIGITAL LOGIC CIRCUITS
Instruction in digital logic circuits covering the 7400 TTL and the CM
series. Codes, registers, counters, memory, combinatorial logic, etc.

HOW TO SET UP A COMPUTER STORE
Guest lectures from owner/operators of computer stores and microco
puter manufacturers. How to become a dealer. How to get a franchi
How to operate a business. How to set up a service facility . . . and more.

KIT BUILDING LAB
Instruction and guidance on kit building. Bring your kits! SPECIA
Anyone taking delivery on a TDL product during the course (must
ordered in advance) will receive this workshop FREE.

For information and reservations contact: Z-80 Seminars, Office of Cont
uing Education, Trenton State College, Trenton, NJ 08625.

A PAYROLL PROGRAM FOR YOUR SMALL BUSINESS

by Robert G. Forbes

A good payroll system is one of the main building blocks of modern computerized business. Until very recently, however, its benefits were only available to large corporations—or small ones willing to deal with an outside service bureau. The advent of the personal computer, however, has changed all that radically. With the microcomputer's flexibility and adaptability to almost limitless specific tasks within the framework of the business world, it now becomes cost-effective for even small organizations with only half a dozen employees or so to computerize their payroll programs.

All one needs is the software. The following payroll system is not hot off the drawing board. It has been up and running for almost a year. Debugged and clean, it's ready to use.

Systems, of course, vary with the hardware at hand. The equipment used in this particular case was:

1. Altair 8800B
2. 32K memory (Altair)
3. 300K disk (Altair)
4. CRT (Lear Siegler)
5. DECwriter printer.

The major concept throughout the design of this system has been to derive a package which meets the following guidelines:

1. Modularity
2. Simplicity
3. Small amount of operator intervention
4. Ease of expansion
5. Reports which ease employer's paperwork.

The reasons I chose to use the above five guidelines are:

1. Modularity—
   The greatest advantage in keeping the flow of programs in a modular format is the ease of maintenance at a later date (if desired) and, operationally, the ability to stop the flow at the end of a module and cross-foot totals, and then resume at a later date.

2. Simplicity—
   Keeping the modules simple in design allows for ease of customization within each module.

3. Small amount of operator intervention—
   A very simple rule of logic applies here. . . . the least amount of intervention results in the least amount of error.

4. Ease of expansion—
   As mentioned in steps one and two, the ability to expand the system as the company grows is of prime importance.

5. Reports which ease employer's paperwork—
   All too often, the problem with using a computerized payroll system is that the employers end up getting bogged down by a lot of unnecessary or rarely-used reports.

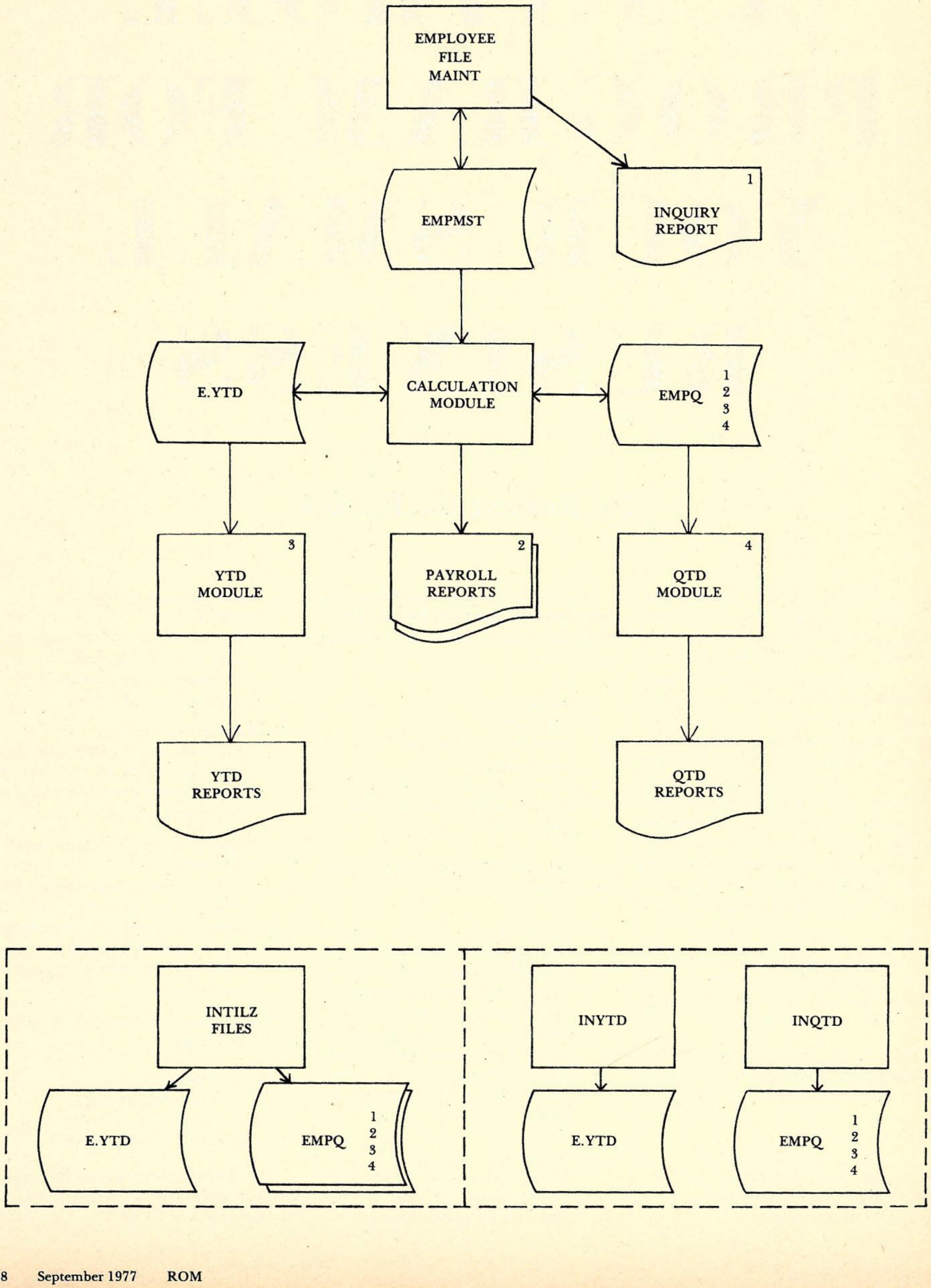
The system is broken up into four major modules:

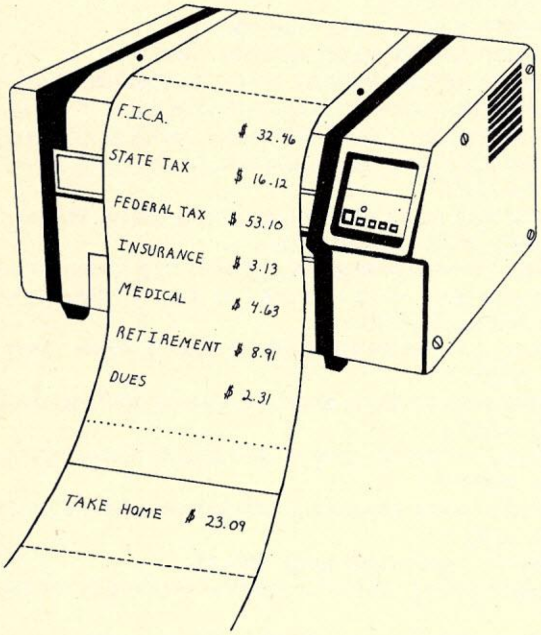
1. File maintenance
2. Payroll calculations
3. Year-to-date posting
4. Quarter-to-date posting.

Detailed delineation of these and further documentation will follow in the next issue of *ROM*. ▼

Illustrated by Joanne Zeiger

EMPLOYEE PAYROLL PROGRAMS FLOWCHART


EMPLOYEE FILE MAINT
EMPMST
1
INQUIRY REPORT
E.YTD
CALCULATION MODULE
EMPQ 1 2 3 4
3
YTD MODULE
2
PAYROLL REPORTS
4
QTD MODULE
YTD REPORTS
QTD REPORTS
INTILZ FILES
E.YTD
EMPQ 1 2 3 4
INYTD
INQTD
E.YTD
EMPQ 1 2 3 4



PAYROLL PROGRAMS

PROGRAM [EMP.1]

```
10 '***** P A Y R O L L  F I L E  M A I N T ****
20 '
30 '      PROGRAM NAME : [EMP.1]
40 '      VERSION      : 1.2
50 '      COPYRIGHT    : 1977
60 '      AUTHOR       : ROBERT G. FORBES
70 '
80 OPEN "R",1,"[EMPMST]",0
90 FIELD#1,20 AS NM$,20 AS AD$,20 AS CT$,4 AS ST$,1 AS MS$,5
AS Z$,4 AS W$,4 AS D$,1 AS S1$,1 AS S2$,4 AS S3$,4 AS S4$,1 AS S5$,
4 AS EN$,11 AS SJ$
100 PRINTCHR$(26)
110 PRINT"ENTER ONE OF THE FOLLOWING SELECTIONS...."
120 PRINT:PRINT
130 PRINT" 'ADD'      - - - ADD A NEW EMPLOYEE TO THE
FILE"
140 PRINT" 'CHANGE' - - - CHANGE EMPLOYEE
INFORMATION"
150 PRINT" 'EXAMINE'- - - EXAMINE EMPLOYEE FILE"
160 PRINT" 'LIST'     - - - LIST EMPLOYEE'S AND
EMPLOYEE'S NUMBERS"
170 PRINT"'RETURN' - - - RETURNTO OPERATINGSYSTEM"
180 PRINT" 'EMP.2'    - - - EXECUTE PHASE 2"
190 PRINT:PRINT
200 SEL$="":INPUT "SELECTION ";SEL$
210 PRINTCHR$(26)
220 IF LEFT$(SEL$,3) = "ADD"THEN290
230 IF LEFT$(SEL$,3) = "CHA"THEN520
240 IF LEFT$(SEL$,3) = "EXA"THEN800
250 IF LEFT$(SEL$,3) = "LIS"THEN920
260 IF LEFT$(SEL$,3) = "RET"THEN1000
270 IF LEFT$(SEL$,5) = "EMP.2"THEN CLOSE:LOAD"[EMP.2]",
0,R
280 GOTO100
290 REM***** ADD ROUTINE *****
300 INPUT "EMPLOYEE NUMBER ";EN
310 INPUT "NAME ";NN$
320 INPUT "STREET ";SE$
330 INPUT "CITY ";CC$
340 INPUT "STATE ";SS$
350 INPUT "ZIP ";ZZ$
360 INPUT "SOCIAL SECURITY NUMBER ";SR$:LSET SJ$=SR$
370 INPUT "WEEKLY GROSS ";WG
380 INPUT "DEDUCTIONS (INSUR) ";DD
390 INPUT "MARITAL STATUS (M OR S ) ";M$
400 INPUT "WEEK OR MONTH PAY (W OR M) ";SW$:LSET
S1$=SW$
410 INPUT "NUM OF DED ";SW$:LSET S2$=SW$
420 INPUT "HOURS PER WEEK ";S3:LSET S3$=MKS$(S3)
430 INPUT "RATE PER HOUR ";S4:LSET S4$=MKS$(S4)
440 INPUT "SWITCH 5 ";SW$:LSET S5$=SW$
450 LSET NM$=NN$:LSET AD$=SE$:LSET CT$=CC$:LSET
ST$=SS$:LSET Z$=ZZ$
460 LSET W$=MKS$(WG):LSET D$=MKS$(DD):LSET MS$=M$
:LSET EN$=MKS$(EN)
470 PUT #1,EN
480 INPUT "ANY MORE ADDITIONS ";ANS$
490 IF LEFT$(ANS$,1) = "Y"THENPRINTCHR$(26):GOTO290
500 OPEN "O",2,"[EMPKEY]",0:PRINT#2,EN:CLOSE2
510 GOTO100
520 REM***** CHANGE ROUTINE ****
530 INPUT "EMPLOYEE NUMBER ";EN
540 GET#1,EN
550 GOSUB1070
560 INPUT "ENTER FIELD NUMBER YOU WISH TO CORRECT ";
NN
570 IF NN<1 OR NN>14 THEN 560
580 ON NN GOTO 620,630,640,650,660,670,680,690,700,710,720,730,
740,750
590 GOTO560
600 PRINTCHR$(26):GOSUB1070
610 INPUT "HIT THE RETURN KEY WHEN READY";ANS$:GOTO
780
620 INPUT "ENTER CHANGED NAME ";NN$:LSETNM$=NN$:
GOTO760
630 INPUT "ENTER CHANGED STREET ";SS$:LSET AD$=SS$:
GOTO760
640 INPUT "ENTER CHANGED CITY ";CC$:LSET CT$=CC$:
GOTO760
650 INPUT "ENTER CHANGED STATE ";SS$:LSET ST$=SS$:
GOTO760
660 INPUT "ENTER CHANGED ZIP ";ZZ$:LSET Z$=ZZ$:GOTO
760
670 INPUT "ENTER CHANGED WK-GROSS ";WG:LSET W$=
MKS$(WG):GOTO760
680 INPUT "ENTER CHANGED DEDUCTIONS ";DD:LSET D$=
MKS$(DD):GOTO760
690 INPUT "ENTER CHANGED MARITAL STATUS ";M$:LSET
MS$=M$:GOTO760
700 INPUT "ENTER CHANGED WK OR MNTH ";SW$:LSET S1$=
SW$:GOTO760
710 INPUT "ENTER CHANGED NUM OF DED ";SW$:LSET S2$=
SW$:GOTO760
720 INPUT "ENTER CHANGED HOURS ";SW:LSET S3$=MKS$
(SW):GOTO760
730 INPUT "ENTER CHANGED PAY-RATE(PER-HOUR) ";SW:
LSET S4$=MKS$(SW):GOTO760
740 INPUT "ENTER CHANGED SWITCH 5 ";SW$:LSET S5$=SW$
:GOTO760
750 INPUT "ENTER CHANGED SOCIAL SECURITY NUMBER ";
```

```
SR$: LSETSJ$ = SR$: GOTO760
760 PUT #1,EN
770 GOTO 600
780 INPUT “ANY MORE CHANGES ”;ANS$: IF LEFT$(ANS$,1) =
“N”THEN100
790 GOTO 560
800 REM******* EXAMINE ROUTINE ********
810 INPUT “EMPLOYEE NUMBER ”;EN
820 INPUT “DO YOU WISH TO PRINT REPORT ”;PT$
830 IF LEFT$(PT$,1) = “Y”THEN CONSOLE 18,1
840 ON ERROR GOTO 900
850 GET#1,EN
860 PRINTCHR$(26)
870 GOSUB1070
880 IF LEFT$(PT$,1) = “Y”THENCONSOLE16,1
890 PRINT: PRINT: INPUT“HIT THE RETURN KEY WHEN
READY”;ANS$: GOTO100
900 PRINT“SORRY EMPLOYEE NUMBER ”;EN;” DOES NOT
EXIST”
910 RESUME 880
920 REM*** LIST ROUTINE ****
930 OPEN “I”,2,”[EMPKEY]”,0: INPUT#2,EMAX: CLOSE2
940 PRINT“EMPLOYEE NUMBER          NAME”
950 FOR I=1TOEM
960 GET #1,I
970 PRINT TAB(5)I;TAB(22)NM$
980 NEXT I
990 INPUT “HIT RETURN WHEN READY”;ANS$: GOTO100
1000 REM*** EXIT ROUTINE ***
1010 UNLOAD
1020 PRINT“PLEASE REMOVE DISKETTE AND MOUNT SYSTEM
DISKETTE”
1030 INPUT “HIT RETURN KEY WHEN READY”
1040 FOR I=1TO2000: NEXTI
1050 MOUNT
1060 LOAD“OP-SYS”,0,R
1070 REM***** PRINT EMPLOYEE INFORMATION *****
1080 PRINTCHR$(26) : PRINTTAB(20) “EMPLOYEE NUMBER ”;
EN
1090 PRINT: PRINT: PRINT
1100 PRINTTAB(7) “1.-NAME..........”;TAB(25)NM$
1110 PRINTTAB(7) “2.—STREET........ ”;TAB(25)AD$
1120 PRINTTAB(7) “3.—CITY........... ”;TAB(25)CT$
1130 PRINTTAB(7) “4.—STATE......... ”;TAB(25)ST$
1140 PRINTTAB(7) “5.—ZIP.............”;TAB(25)Z$
1150 PRINTTAB(7) “6.—WK-GROSS..... ”;TAB(25)CVS(W$)
1160 PRINTTAB(7) “7.—DEDUCT.......  ”;TAB(25)CVS(D$)
1170 PRINTTAB(7) “8.—MAR.STAT.....  ”;TAB(25)MS$
1180 PRINTTAB(7) “9.—WK OR MN.....  ”;TAB(25)S1$
1190 PRINTTAB(7) “10—# OF DED.......”;TAB(25)S2$
1200 PRINTTAB(7) “11—HOURS WKD...  ”;TAB(25)CVS(S3$)
1210 PRINTTAB(7) “12—PAY-RATE(HR).”;TAB(25)CVS(S4$)
1220 PRINTTAB(7) “13—SWITCH 5...... ”;TAB(25)S5$
1230 PRINTTAB(7) “14—SOC.SEC.......  ”;TAB(25)SJ$
1240 RETURN
```

PROGRAM [EMP.1] IS 124 LINES LONG

PROGRAM [EMP.2]

```
10 ' ***** P A Y R O L L  FILE UPDATES *****
20 '
30 '        PROGRAM NAME : [EMP.2]
40 '        VERSION      : 1.2
50 '        COPYRIGHT    : 1977
60 '        AUTHOR       : ROBERT G. FORBES
70 '
80 DEF FNRN(X) = (X + 5E—03) *100
90 DEF FND(X) = (INT(FNRN(X)) /100)
100 PRINTCHR$(26)
110 INPUT “ENTER DATE ”;DATE$
120 INPUT “ENTER WEEK NUMBER ”; WN
130 INPUT “IS THIS REPORT TO GO TO THE PRINTER ”;NA$
140 IF LEFT$(NA$,1) = “Y”THEN CONSOLE18,1
150 DIM S1(7),S2(7),S3(7),S4(7),M1(7),M2(7),M3(7),M4(7)
160 DIM H1(7),H2(7),H3(7),H4(7),G1(7),G2(7),G3(7),G4(7)
170 DIM EM(8)
180 DIM NA$(10),TG(10),TD(10),TT(10),TF(10),TN(10)
190 OPEN “R”,1,“[EMPMST]”,0
200 FIELD#1,20 AS NM$,20 AS AD$,20 AS CT$,4 AS ST$,1 AS MS$,5
AS Z$,4 AS W$,4 AS D$,1 AS S1$,1 AS S2$,4 AS S3$,4 AS S4$,1 AS
S5$,4 AS EN$,11 AS SJ$
210 FOR I=1TO7: READ S1(I) : READS2(I) : READ S3(I) : READ
S4(I) : NEXT I
220 FOR I=1TO7: READ M1(I) : READM2(I) : READM3(I) : READ
M4(I) : NEXT I
230 FOR I=1TO7: READ H1(I) : READH2(I) : READH3(I) : READ
H4(I) : NEXT I
240 FOR I=1TO7: READ G1(I) : READG2(I) : READG3(I) : READ
G4(I) : NEXT I
250 FOR I=1TO8: READ EM(I) : NEXTI
260 DATA 25,67,0,16,67,115,6.72,20,115,183,16.32,23,183,240,31.96,
21
270 DATA 240,279,43.93,26,279,346,54.07,30,346,999,74.17,36
280 DATA 48,96,0,17,96,173,8.16,20,173,264,23.56,17,264,346,
39.03,25
290 DATA 346,433,59.53,28,433,500,83.89,32,500,999,105.33,36
300 DATA 108,292,0,16,292,500,29.44,20,500,792,71.04,23,792,1042,
138.20
310 DATA 21,1042,1208,190.70,26,1208,1500,233.86,30,1500,9999,
321.46,36
320 DATA 208,417,0,17,417,750,35.53,20,750,1146,102.13,17,1146,
1500
330 DATA 169.45,25,1500,1875,257.95,28,1875,2167,362.95,32,2167,
9999,456.39,36
340 DATA 14.40,28.8,31.3,62.5,187.5,375.0,750.0,2.1
350 OPEN “I”,2,“[EMPKEY]”,0: INPUT#2,MAX: CLOSE2
360 IF WN< 52 THENWM=4
370 IF WN< 39THENWM=3
380 IF WN<26THENWM=2
390 IF WN<13THENWM=1
400 IF WN<1THENSTOP
410 IF WN >52THENSTOP
420 QF$= “[EMPQ“+ STR$(WM) + ”] ”
430 FOR PR=1TOMAX
440 GET #1,PR
450 HR=CVS(S3$) : PH=CVS(S4$) : NE= VAL(S2$)
460 PRINT “— — — — — — — — — — — — — — — — — — — — — —
— — — — — — — — — — — — — — — — — — — — — — — —
— — — — — — — — — — —”
470 PRINT TAB(20) “DATE ”; DATE$
480 PRINT TAB(20) “EMPLOYEE NUMBER ”; PR
490 PRINT: PRINT
500 PRINT “EMPLOYEE : ”;NM$
510 PRINT “SEC.SEC. : ”;SJ$
520 PRINT “HR'S= ”;HR; “ NUM OF DED = ”;NE; “ PAY RATE = ”
;PH
530 W=CVS(W$)
540 IF HR< > 0THENW= (HR*PH)
550 PRINT “GROSS : ”; W
560 IF S1$= “W”THENA1 = 1ELSE A1 = 4
570 LET D= (EM(A1) *NE) : D=FND(D)
580 N1 = W—D
```

```
590 IF MS$ = "S"THEN GOSUB810
600 IF MS$ = "M"THENGOSUB920
610 Y = T1 + R: Y = FND(Y)
620 FI = W*5.85/100: FI = FND(FI)
630 D3 = CVS(D$) : D3 = FND(D3)
640 N3 = (W - Y - D3 - FI) : N3 = FND(N3)
650 PRINT "TOTAL DEDUCTIONS TO BE WITHHELD...";D3
660 PRINT "TOTAL TAX TO BE WITHHELD............";Y
670 PRINT "TOTAL FICA WITHHELD..................";FI
680 PRINT "TOTAL NET ..............................";N3
690 PRINT "- - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - -"
700 PRINT: PRINT: PRINT
710 NA$(PR) = SJ$
720 TG(PR) = W
730 TD(PR) = D3
740 TT(PR) = Y
750 TF(PR) = FI
760 TN(PR) = N3
770 NEXT PR
780 GOTO1230
790 IF LEFT$(NA$,1) = "Y"THENCONSOLE16,1
800 END
810 REM****** SINGLE TAX LOOKUP ******
820 IF S1$ = "M"THEN1030
830 FOR I = 1TO7
840 IF W > S1(I) THEN870
850 NEXT I
860 PRINT "ERROR IN TAX LOOKUP- (SINGLE)": GOTO790
870 IF W > S2(I) THEN 850
880 T1 = S3(I) : P1 = S4(I)
890 E1 = (W - S1(I)) *S4(I) /100
900 R = ((N1 - S1(I)) *P1) /100
910 RETURN
920 REM****** MARRIED TAX LOOKUP *******
930 IF S1$ = "M"THEN1130
940 FOR I = 1TO7
950 IF W > M1(I) THEN980
960 NEXT I
970 PRINT "ERROR IN TAX LOOKUP- (MARRIED)": GOTO790
980 IF CVS(W$) > M2(I) THEN960
990 T1 = M3(I) : P1 = M4(I)
1000 E1 = (W - M1(I)) *M4(I) /100
1010 R = ((N1 - M1(I)) *P1) /100
1020 RETURN
1030 REM****** SINGLE MONTHLY TAX LOOKUP *****
1040 FOR I = 1TO7
1050 IF W > H1(I) THEN1080
1060 NEXT I
1070 PRINT "ERROR IN TAX LOOKUP- (SINGLE-MONTHLY)"
: GOTO790
1080 IF W > H2(I) THEN1060
1090 T1 = H3(I) : P1 = H4(I)
1100 E1 = (W - H1(I)) *H4(I) /100
1110 R = ((N1 - H1(I)) *P1) /100
1120 RETURN
1130 REM****** MARRIED MONTHLY TAX LOOKUP *****
1140 FOR I = 1TO7
1150 IF W > G1(I) THEN1180
1160 NEXT I
1170 PRINT "ERROR IN TAX LOOKUP- (MARRIED-MONTHLY)
": GOTO790
1180 IF W > G2(I) THEN1160
1190 T1 = G3(I) : P1 = G4(I)
1200 E1 = (W - G1(I)) *G4(I) /100
1210 R = ((N1 - G1(I)) *P1) /100
1220 RETURN
1230 PRINT TAB(21) "DATE ";DATE$
1240 PRINT TAB(16) "EMPLOYEE PAYROLL WEEK NUMBER ";
WN
1250 PRINT: PRINT
1260 PRINT "SOC.SEC.    GROSS    DED    TAX    FICA    NET"
1270 T1 = 0: T2 = 0: T3 = 0: T4 = 0: T5 = 0
1280 FOR I = 1TOPR - 1
1290 PRINT NA$(I);
1300 PRINT TAB(13) TG(I);
1310 T1 = T1 + TG(I)
1320 PRINT TAB(22) TD(I);
1330 T2 = T2 + TD(I)
1340 PRINT TAB(35) TT(I);
1350 T3 = T3 + TT(I)
1360 PRINT TAB(44) TF(I);
1370 T4 = T4 + TF(I)
1380 PRINT TAB(55) TN(I);
1390 T5 = T5 + TN(I)
1400 PRINT
1410 NEXT I
1420 PRINT "        - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - -"
1430 PRINT "[TOTALS]";
1440 PRINT TAB(13) T1;
1450 PRINT TAB(22) T2;
1460 PRINT TAB(35) T3;
1470 PRINT TAB(44) T4;
1480 PRINT TAB(55) T5;
1490 T2 = T2 + TD(I)
1500 PRINT
1510 CLOSE
1520 OPEN "R",3,"[E.YTD]",0
1530 FIELD #3,4ASG1$,4ASG2$,4ASD1$,4ASD2$,4AST1$,4AST2$,
4ASF1$,4ASF2$,4ASN1$,4ASN2$,11ASSS$
1540 OPEN "R",4,QF$
1550 FIELD#4,4ASQY$,4ASDY$,4ASTY$,4ASFY$,4ASNY$,11ASSY$
1560 FOR I = 1TOPR - 1
1570 GET #3,I
1580 GET #4,I
1590 LSET G1$ = MKS$(TG(I))
1600 X = CVS(G2$) : X = X + TG(I)
1610 LSET G2$ = MKS$(X)
1620 XX = CVS(QY$) : LSETQY$ = MKS$(XX + TG(I))
1630 LSET D1$ = MKS$(TD(I))
1640 X = CVS(D2$) : X = X + TD(I)
1650 LSET D2$ = MKS$(X)
1660 XX = CVS(DY$) : LSETDY$ = MKS$(XX + TD(I))
1670 LSET T1$ = MKS$(TT(I))
1680 X = CVS(T2$) : X = X + TT(I)
1690 LSET T2$ = MKS$(X)
1700 XX = CVS(TY$) : LSETTY$ = MKS$(XX + TT(I))
1710 LSET F1$ = MKS$(TF(I))
1720 X = CVS(F2$) : X = X + TF(I)
1730 LSET F2$ = MKS$(X)
1740 XX = CVS(FY$) : LSETFY$ = MKS$(XX + TF(I))
1750 LSET N1$ = MKS$(TN(I))
1760 X = CVS(N2$) : X = X + TN(I)
1770 LSET N2$ = MKS$(X)
1780 XX = CVS(NY$) : LSETNY$ = MKS$(XX + TN(I))
1790 LSET SS$ = NA$(I)
1800 LSET SY$ = NA$(I)
1810 PUT #3,I
1820 PUT #4,I
1830 NEXT I
1840 CLOSE
1850 CONSOLE 16,1
1860 PRINT: PRINT
1870 PRINT"*** PLEASE NOTE....": PRINT
1880 PRINT "        YTD EMPLOYEE FILE HAS BEEN UPDATED"
```

```
1890 PRINT "          QUARTER FILE ";QF$;" HAS BEEN
UPDATED"
1900 OPEN "0",4,"[EMPWK]",0:PRINT#4,WN:CLOSE
1910 LOAD"[EMP.3]",0,R
```

PROGRAM [EMP.2] IS 316 LINES LONG

PROGRAM [EMP.3]

```
10 ' ***** PAYROLL Y.T.D. REPORTS *****
20 '
30 '        PROGRAM NAME : [EMP.3]
40 '        VERSION      : 1.2
50 '        COPYRIGHT    : 1977
60 '        AUTHOR       : ROBERT G.FORBES
70 '
80 OPEN "I",1,"[EMPWK]",0:INPUT#1,WEEK$:CLOSE1
90 OPEN "I",2,"[EMPKEY]",0:INPUT#2,EMAX:CLOSE2
100 OPEN "R",3,"[E.YTD]",0
110 FIELD#3,4ASG1$,4ASG2$,4ASD1$,4ASD2$,4AST1$,4AST2$,
4ASF1$,4ASF2$,4ASN1$,4ASN2$,11 AS SS$
120 INPUT "IS THIS REPORT TO GO TO THE PRINTER ";ANS$
130 INPUT "HIT RETURN WHEN READY";RES$
140 IF LEFT$(ANS$,1) = "Y"THEN CONSOLE 18,1
150 PRINT TAB(20)"THE COMPUTER STORE"
160 PRINT TAB(14)"Y.T.D.FIGURES WEEK NUMBER ";WEEK$
170 FOR I=1TOEMAX
180 GET #3,I
190 PRINT "SS NUMBER ";SS$
200 PRINT"WEEK GROSS ";CVS(G1$),"YTD GROSS ";CVS(G2$)
210 PRINT"WEEK DED ";CVS(D1$),"YTD DED ";CVS(D2$)
220 PRINT"WEEK TAX ";CVS(T1$),"YTD TAX ";CVS(T2$)
230 PRINT"WEEK FICA ";CVS(F1$),"YTD FICA ";CVS(F2$)
240 PRINT"WEEK NET ";CVS(N1$),"YTD NET ";CVS(N2$)
250 IF LEFT$(ANS$,1) = "Y"THEN 270
260 INPUT "HIT THE RETURN KEY WHEN READY";ANS$
270 PRINT:PRINT:PRINT
280 NEXT I
290 IF LEFT$(ANS$,1) = "Y"THENCONSOLE 16,1
300 CLOSE
310 LOAD"[EMP.4]",0,R
```

PROGRAM [EMP.3] IS 348 LINES LONG

PROGRAM [EMP.4]

```
10 ' ****** PAYROLL QUARTER REPORTS *****
20 '
30 '        PROGRAM NAME : [EMP.4]
40 '        VERSION      : 1.2
50 '        COPYRIGHT    : 1977
60 '        AUTHOR       : ROBERT G. FORBES
70 '
80 ON ERROR GOTO410
90 PRINTCHR$(26)
100 INPUT "ENTER QUARTER YOU WISH TO PRINT (1-4) ";Q
110 IF Q<1 OR Q>4THEN100
120 DF$="[EMPQ"+STR$(Q) + "]"
130 OPEN "R",1,DF$,0
140 OPEN "I",2,"[EMPKEY]",0:INPUT#2,EMAX:CLOSE2
150 INPUT "READY PRINTER- - -HIT RETURN WHEN
READY";REP$
160 CONSOLE 18,1
170 PRINT:PRINTTAB(25)"QUARTER FIGURES FOR QTR ";Q
180 PRINT:PRINT
190 FIELD#1,4AS QY$,4ASDY$,4ASTY$,4ASFY$,4ASNY$,11ASSY$
200 FOR I=1TOEMAX
210 GET #1,I
220 Q=CVS(QY$):D=CVS(DY$):T=CVS(TY$):F=CVS(FY$)
:N=CVS(NY$)
230 PRINT "SOCIAL SECURITY NUMBER  ";SY$
240 PRINT "QUARTER TO DATE GROSS     ";Q
250 PRINT "QUARTER TO DATE DEDUCTIONS ";D
260 PRINT "QUARTER TO DATE TAXS       ";T
270 PRINT "QUARTER TO DATE FICA       ";F
280 PRINT "QUARTER TO DATE NET        ";N
290 TQ=TQ+Q:TD=TD+D:TT=TT+T:TF=TF+F:TN=
TN+N
300 PRINT:PRINT:PRINT
310 NEXT I
320 CLOSE
330 FOR I=1TO5:PRINT:NEXT
340 PRINTTAB(25)"TOTAL Q.T.D. FOR ALL EMPLOYEES"
350 PRINT:PRINT
360 PRINT"TOTAL GROSS       ";TQ
370 PRINT"TOTAL DEDUCTIONS ";TD
380 PRINT"TOTAL TAXS        ";TT
390 PRINT"TOTAL FICA        ";TF
400 PRINT"TOTAL NET         ";TN
410 CONSOLE 16,1
420 PRINT CHR$(26)
430 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
440 PRINTTAB(10) "END OF PAYROLL PROCESSING"
450 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
460 NEW
```

PROGRAM [EMP.4] IS 395 LINES LONG

PROGRAM [INTILZ]

```
10 ' ****** PAYROLL -INTILIZE FILES PROGRAM- ******
20 '
30 '        PROGRAM NAME : [INTILZ]
40 '        VERSION      : 1.2
50 '        COPYRIGHT    : 1977
60 '        AUTHOR       : ROBERT G. FORBES
70 '
80 PRINTCHR$(26)
90 INPUT "DO YOU WISH TO INITILIZE EMPLOYEE YEAR TO
DATE FILE ";ANS$
100 IF LEFT$(ANS$,2) = "YE"THENS9=1ELSE S9=0
110 IF S9=1THEN210
120 PRINTCHR$(26)
130 PRINT "YOU MAY INITILIZE THE FOLLOWING FILES......"
140 PRINT:PRINT"[EMPQ1]  [EMPQ2]  [EMPQ3]  [EMPQ4]"
150 PRINT:PRINT:PRINT
160 PRINT
170 INPUT "ENTER FILE NAME TO INITILIZE...'END' TO
TERMINATE ";DF$
180 IF LEFT$(DF$,3) = "END"THENPRINTCHR$(26):END
190 S9=0
200 GOTO240
210 OPEN "R",3,"[E.YTD]",0
```

```
220 FIELD#3,4ASG1$,4ASG2$,4ASD1$,4ASD2$,4AST1$,4AST2$,
4ASF1$,4ASF2$,4ASN1$,4ASN2$,11ASSS$
230 GOTO260
240 OPEN "R",4,DF$,0
250 FIELD#4,4AS QY$,4ASDY$,4ASTY$,4ASFY$,4ASNY$,11ASSY$
260 Z=0
270 FOR I=1TO20
280 IF S9=0THEN400
290 LSET G1$=MKS$(Z)
300 LSET G2$=MKS$(Z)
310 LSET D1$=MKS$(Z)
320 LSET D2$=MKS$(Z)
330 LSET T1$=MKS$(Z)
340 LSET T2$=MKS$(Z)
350 LSET F1$=MKS$(Z)
360 LSET F2$=MKS$(Z)
370 LSET N1$=MKS$(Z)
380 LSET N2$=MKS$(Z)
390 LSET SS$=""
400 LSET QY$=MKS$(Z):LSET DY$=MKS$(Z):LSET TY$=MKS$
(Z):LSET FY$=MKS$(Z)
410 LSET NY$=MKS$(Z):LSET SY$=MKS$(Z)
420 IF S9=0THEN450
430 PUT #3,I
440 GOTO460
450 PUT #4,I
460 NEXT I
470 CLOSE
480 GOTO120
490 END
```

PROGRAM [INTILZ] IS 445 LINES LONG

PROGRAM [INQTD]

```
10 '****** P A Y R O L L (INITILZE)  QUARTER REPORTS *****
20 '
30 '        PROGRAM NAME : [EMP.4]
40 '        VERSION      : 1.2
50 '        COPYRIGHT    : 1977
60 '        AUTHOR       : ROBERT G. FORBES
70 '
80 PRINTCHR$(26)
90 INPUT "ENTER QUARTER YOU WISH TO INITILZE (1-4) ";Q
100 IF Q<1 OR Q>4THEN90
110 DF$="[EMPQ"+STR$(Q)+"]"
120 OPEN "R",1,DF$,0
130 FIELD#1,4AS QY$,4ASDY$,4ASTY$,4ASFY$,4ASNY$,11ASSY$
140 PRINTCHR$(26)
150 INPUT"ENTER I.D. NUMBER ";I
160 IF I=0THEN270
170 INPUT"ENTER QUARTER GROSS ";G
180 INPUT"ENTER QUARTER DEDUCTIONS ";D
190 INPUT"ENTER QUARTER TAX ";T
200 INPUT"ENTER QUARTER FICA ";F
210 INPUT"ENTER QUARTER NET ";N
220 INPUT"ENTER SOCIAL SECURITY NUMBER ";S$
230 LSET QY$=MKS$(G):LSETDY$=MKS$(D):LSETTY$=MKS
(T):LSETFY$=MKS$(F)
240 LSET NY$=MKS$(N):LSET SY$=S$
250 PUT#1,I
260 GOTO140
270 END
```

PROGRAM [INQTD] IS 473 LINES LONG

SAMPLE RUNS

```
                    EMPLOYEE NUMBER 1

1.-NAME.............. J. JONES
2.-STREET............ 123 MAPLE AVE
3.-CITY.............. TIMBUKTU
4.-STATE............. CT
5.-ZIP............... 06000
6.-WK-GROSS.......... 210.5
7.-DEDUCT............ 8.66
8.-MAR.STAT.......... M
9.-WK OR MN.......... W
10-# OF DED.......... 0
11-HOURS WKD......... 0
12-PAY-RATE(HR)...... 0
13-SWITCH 5.......... 0
14-SOC.SEC........... 000-00-0001

                    EMPLOYEE NUMBER 2

1.-NAME.............. S. SMITH
2.-STREET............ 678 ADAMS ROAD
3.-CITY.............. TIMBUKTU
4.-STATE............. CT
5.-ZIP............... 06000
6.-WK-GROSS.......... 187.94
7.-DEDUCT............ 5.88
8.-MAR.STAT.......... M
9.-WK OR MN.......... W
10-# OF DED.......... 0
11-HOURS WKD......... 0
12-PAY-RATE(HR)...... 0
13-SWITCH 5.......... 0
14-SOC.SEC........... 000-00-0002

                    EMPLOYEE NUMBER 3

1.-NAME.............. M. JONES
2.-STREET............ 17 PUTNAM STREET
3.-CITY.............. TIMBUKTU
4.-STATE............. CT
5.-ZIP............... 06000
6.-WK-GROSS.......... 0
7.-DEDUCT............ 3.66
8.-MAR.STAT.......... S
9.-WK OR MN.......... W
10-# OF DED.......... 0
11-HOURS WKD......... 40
12-PAY-RATE(HR)...... 3.75
13-SWITCH 5.......... 0
14-SOC.SEC........... 000-00-0003

                    EMPLOYEE NUMBER 4

1.-NAME.............. D. SMITH
2.-STREET............ 89 WINDSOR STREET
3.-CITY.............. TIMBUKTU
4.-STATE............. CT
5.-ZIP............... 06000
6.-WK-GROSS.......... 0
7.-DEDUCT............ 0
8.-MAR.STAT.......... S
9.-WK OR MN.......... W
10-# OF DED.......... 0
11-HOURS WKD......... 20
12-PAY-RATE(HR)...... 3.15
13-SWITCH 5.......... 0
14-SOC.SEC........... 000-00-0004
```

```
------------------------------------------------------------------


                   DATE MAY 13, 1977
                  EMPLOYEE NUMBER 1

EMPLOYEE : J.JONES
SEC.SEC. : 000-00-0001
HR'S= 0 NUM OF DED = 0 PAY RATE = 0
GROSS : 210.5
TOTAL DEDUCTIONS TO BE WITHHELD...          8.66
TOTAL TAX TO BE WITHHELD...........        29.94
TOTAL FICA WITHHELD..................      12.31
TOTAL NET.............................    159.59

                   DATE MAY 13, 1977
                  EMPLOYEE NUMBER 2

EMPLOYEE : S. SMITH
SEC.SEC. : 000-00-0002
HR'S= 0 NUM OF DED = 0 PAY RATE = 0
GROSS : 187.94
TOTAL DEDUCTIONS TO BE WITHHELD...          5.88
TOTAL TAX TO BE WITHHELD...........         26.1
TOTAL FICA WITHHELD..................      10.99
TOTAL NET.............................    144.97

                   DATE MAY 13, 1977
                  EMPLOYEE NUMBER 3

EMPLOYEE : M. JONES
SEC.SEC. : 000-00-0003
HR'S= 40 NUM OF DED = 0 PAY RATE = 3.75
GROSS : 150
TOTAL DEDUCTIONS TO BE WITHHELD...          3.66
TOTAL TAX TO BE WITHHELD...........        24.37
TOTAL FICA WITHHELD..................       8.78
TOTAL NET.............................    113.19

                   DATE MAY 13, 1977
                  EMPLOYEE NUMBER 4

EMPLOYEE : D. SMITH
SEC.SEC. : 000-00-0004
HR'S= 20 NUM OF DED = 0 PAY RATE = 3.15
GROSS : 63
TOTAL DEDUCTIONS TO BE WITHHELD...          0
TOTAL TAX TO BE WITHHELD...........         6.08
TOTAL FICA WITHHELD..................       3.69
TOTAL NET.............................     53.23

                   DATE MAY 13, 1977
           EMPLOYEE PAYROLL WEEK NUMBER 1

SOC.SEC.        GROSS    DED     TAX     FICA    NET
000-00-0001     210.5    8.66    29.94   12.31   159.59
000-00-0002     187.94   5.88    26.1    10.99   144.97
000-00-0003     150      3.66    24.37   8.78    113.19
000-00-0004     63       0       6.08    3.69    53.23
[TOTALS]        611.44   18.2    86.49   35.77   470.98

                   DATE MAY 20, 1977
                  EMPLOYEE NUMBER 1

EMPLOYEE : J. JONES
SEC.SEC. : 000-00-0001
HR'S= 0 NUM OF DED = 0 PAY RATE = 0
GROSS : 210.5
TOTAL DEDUCTIONS TO BE WITHHELD...          8.66
TOTAL TAX TO BE WITHHELD...........        29.94
TOTAL FICA WITHHELD..................      12.31
TOTAL NET.............................    159.59

                   DATE MAY 20, 1977
                  EMPLOYEE NUMBER 2

EMPLOYEE : S. SMITH
SEC.SEC. : 000-00-0002
HR'S= 0 NUM OF DED = 0 PAY RATE = 0
GROSS : 187.94
TOTAL DEDUCTIONS TO BE WITHHELD...          5.88
TOTAL TAX TO BE WITHHELD...........         26.1
TOTAL FICA WITHHELD..................      10.99
TOTAL NET.............................    144.97

                   DATE MAY 20, 1977
                  EMPLOYEE NUMBER 3

EMPLOYEE : M. JONES
SEC.SEC. : 000-00-0003
HR'S= 40 NUM OF DED = 0 PAY RATE = 3.75
GROSS : 150
TOTAL DEDUCTIONS TO BE WITHHELD...          3.66
TOTAL TAX TO BE WITHHELD...........        24.37
TOTAL FICA WITHHELD..................       8.78
TOTAL NET.............................    113.19

                   DATE MAY 20, 1977
                  EMPLOYEE NUMBER 4

EMPLOYEE : D. SMITH
SEC.SEC. : 000-00-0004
HR'S= 20 NUM OF DED = 0 PAY RATE = 3.15
GROSS : 63
TOTAL DEDUCTIONS TO BE WITHHELD...          0
TOTAL TAX TO BE WITHHELD...........         6.08
TOTAL FICA WITHHELD..................       3.69
TOTAL NET.............................     53.23

                   DATE MAY 20, 1977
           EMPLOYEE PAYROLL WEEK NUMBER 2

SOC.SEC.        GROSS    DED     TAX     FICA    NET
000-00-0001     210.5    8.66    29.94   12.31   159.59
000-00-0002     187.94   5.88    26.1    10.99   144.97
000-00-0003     150      3.66    24.37   8.78    113.19
000-00-0004     63       0       6.08    3.69    53.23
[TOTALS]        611.44   18.2    86.49   35.77   470.98


------------------------------------------------------------------


                 THE COMPUTER STORE
             Y.T.D.FIGURES WEEK NUMBER 1

SS NUMBER 000-00-0001
WEEK GROSS 210.5    YTD GROSS 210.5
WEEK DED      8.66  YTD DED      8.66
WEEK TAX     29.94  YTD TAX     29.94
WEEK FICA    12.31  YTD FICA    12.31
WEEK NET    159.59  YTD NET    159.59


SS NUMBER 000-00-0002
WEEK GROSS 187.94   YTD GROSS 187.94
WEEK DED      5.88  YTD DED      5.88
WEEK TAX     26.1   YTD TAX     26.1
WEEK FICA    10.99  YTD FICA    10.99
WEEK NET    144.97  YTD NET    144.97
```

```
SS NUMBER 000–00–0003
WEEK GROSS 150      YTD GROSS 150
WEEK DED      3.66  YTD DED      3.66
WEEK TAX     24.37  YTD TAX     24.37
WEEK FICA     8.78  YTD FICA     8.78
WEEK NET    113.19  YTD NET    113.19


SS NUMBER 000–00–0004
WEEK GROSS 63       YTD GROSS 63
WEEK DED   0        YTD DED   0
WEEK TAX    6.08    YTD TAX     6.08
WEEK FICA   3.69    YTD FICA    3.69
WEEK NET   53.23    YTD NET    53.23


              THE COMPUTER STORE
        Y.T.D.FIGURES WEEK NUMBER 2

SS NUMBER 000–00–0001
WEEK GROSS 210.5    YTD GROSS 421
WEEK DED      8.66  YTD DED     17.32
WEEK TAX     29.94  YTD TAX     59.88
WEEK FICA    12.31  YTD FICA    24.62
WEEK NET    159.59  YTD NET    319.18


SS NUMBER 000–00–0002
WEEK GROSS 187.94   YTD GROSS 375.88
WEEK DED      5.88  YTD DED     11.76
WEEK TAX     26.1   YTD TAX     52.2
WEEK FICA    10.99  YTD FICA    21.98
WEEK NET    144.97  YTD NET    289.94


SS NUMBER 000–00–0003
WEEK GROSS 150      YTD GROSS 300
WEEK DED      3.66  YTD DED      7.32
WEEK TAX     24.37  YTD TAX     48.74
WEEK FICA     8.78  YTD FICA    17.56
WEEK NET    113.19  YTD NET    226.38


SS NUMBER 000–00–0004
WEEK GROSS 63       YTD GROSS 126
WEEK DED   0        YTD DED   0
WEEK TAX    6.08    YTD TAX     12.16
WEEK FICA   3.69    YTD FICA     7.38
WEEK NET   53.23    YTD NET    106.46


--------------------------------------------------------------------


             QUARTER FIGURES FOR QTR 1

SOCIAL SECURITY NUMBER               000–00–0001
QUARTER TO DATE GROSS                    210.5
QUARTER TO DATE DEDUCTIONS                8.66
QUARTER TO DATE TAXS                     29.94
QUARTER TO DATE FICA                     12.31
QUARTER TO DATE NET                     159.59


SOCIAL SECURITY NUMBER               000–00–0002
QUARTER TO DATE GROSS                   187.94
QUARTER TO DATE DEDUCTIONS                5.88
QUARTER TO DATE TAXS                     26.1
QUARTER TO DATE FICA                     10.99
```

```
QUARTER TO DATE NET                     144.97


SOCIAL SECURITY NUMBER               000–00–0003
QUARTER TO DATE GROSS                    150
QUARTER TO DATE DEDUCTIONS                3.66
QUARTER TO DATE TAXS                     24.37
QUARTER TO DATE FICA                      8.78
QUARTER TO DATE NET                     113.19


SOCIAL SECURITY NUMBER               000–00–0004
QUARTER TO DATE GROSS                     63
QUARTER TO DATE DEDUCTIONS                 0
QUARTER TO DATE TAXS                      6.08
QUARTER TO DATE FICA                      3.69
QUARTER TO DATE NET                      53.23


        TOTAL Q.T.D. FOR ALL EMPLOYEES

TOTAL GROSS          611.44
TOTAL DEDUCTIONS      18.2
TOTAL TAXS            86.49
TOTAL FICA            35.77
TOTAL NET            470.98


             QUARTER FIGURES FOR QTR 1

SOCIAL SECURITY NUMBER               000–00–0001
QUARTER TO DATE GROSS                    421
QUARTER TO DATE DEDUCTIONS               17.32
QUARTER TO DATE TAXS                     59.88
QUARTER TO DATE FICA                     24.62
QUARTER TO DATE NET                     319.18


SOCIAL SECURITY NUMBER               000–00–0002
QUARTER TO DATE GROSS                   375.88
QUARTER TO DATE DEDUCTIONS               11.76
QUARTER TO DATE TAXS                     52.2
QUARTER TO DATE FICA                     21.98
QUARTER TO DATE NET                     289.94


SOCIAL SECURITY NUMBER               000–00–0003
QUARTER TO DATE GROSS                    300
QUARTER TO DATE DEDUCTIONS                7.32
QUARTER TO DATE TAXS                     48.74
QUARTER TO DATE FICA                     17.56
QUARTER TO DATE NET                     226.38


SOCIAL SECURITY NUMBER               000–00–0004
QUARTER TO DATE GROSS                    126
QUARTER TO DATE DEDUCTIONS                 0
QUARTER TO DATE TAXS                     12.16
QUARTER TO DATE FICA                      7.38
QUARTER TO DATE NET                     106.46

        TOTAL Q.T.D. FOR ALL EMPLOYEES

TOTAL GROSS         1222.88
TOTAL DEDUCTIONS      36.4
TOTAL TAXS           172.98
TOTAL FICA            71.54
TOTAL NET            941.96
```

The Noisy Channel

VMOS is vrooming into the personal computer field at the end of this year or early in '78. Using the standard MOS technology with wicked V-grooves a few millionths of an inch deep etched into the chip, the VMOS circuits are forty percent more powerful, not to mention some twenty-five to thirty-five percent cheaper.

Current flows up and down in the grooves as well as laterally. VMOS slips more info into the same space and the vertical juice jumps speeds for processing as well. Look for superfast 65,536-bit VMOS memory chips for less than the price of a pizza-with-the-works-to-go.

* * *

Is a PDP-11 a little out of your price range? Well, how about an ersatz one at half price -- from National Semiconductor? Emulation may be the sincerest form of flattery, but what will Digital think about National's new mini that uses PDP-11 software? Then again, is National a step behind what's coming up from Digital?

* * *

Memory again -- this time it's expanding almost exponentially. Now bounding bubbles from Texas Instruments are about to pop onto the market with 1472K. Granted, it's a full year away. Still, at forty-five dollars a shot, that's enough to make Harry Lorraine weep.

* * *

Have you heard the one about the remote inquiry, teletype, CRT, Touchtone-accessible microfiche storage? Coming to the microfield -- 8080 and 6800 software for communication control, priority assignment, indexing, and search. Price looks a bit steep yet -- then again, no more teachers' dirty looks or packing books.

* * *

Advanced hardware hackers should check out the MP 20 analog input micro peripheral from Burr-Brown, International Airport Industrial Park, Tucson AZ 85734. Handles thermocouple outputs and other testy analog data acquisitions directly for your 8085, 8048, Z-80 as well as friendly old SC/MP. Tell 'em ROM sent you -- and send us your results.

* * *

A small start-up company is forming outside of Boston to manufacture A/D devices for the 8080, 6800, Z-80 personal computer market. Sensing devices and control boards to couple your greenhouse humidifier, home heating and lighting systems, as well as letting the cat out at 2:00 A.M. Plans are also afin to have the computer feed your tropical fish. Glub glub.

* * *

Next to speech synthesizers, the pet field for microcomputer development seems to be graphics. And a small revolution will face the displaying of digitized data come the middle of '78, when the new _________ system is added to a prominent personal computer manufacturer's line. The 512 by 512 matrix offers 64K of color hue choice per dot!

* * *

With the Digital Group voice synthesizer beginning its chatter from a 32K memory, keep your eyes open for some new CPUs coming by way of the same firm. Referred to simply as #5 and #6, the as-yet-unnamed CPUs promise great things for next year.

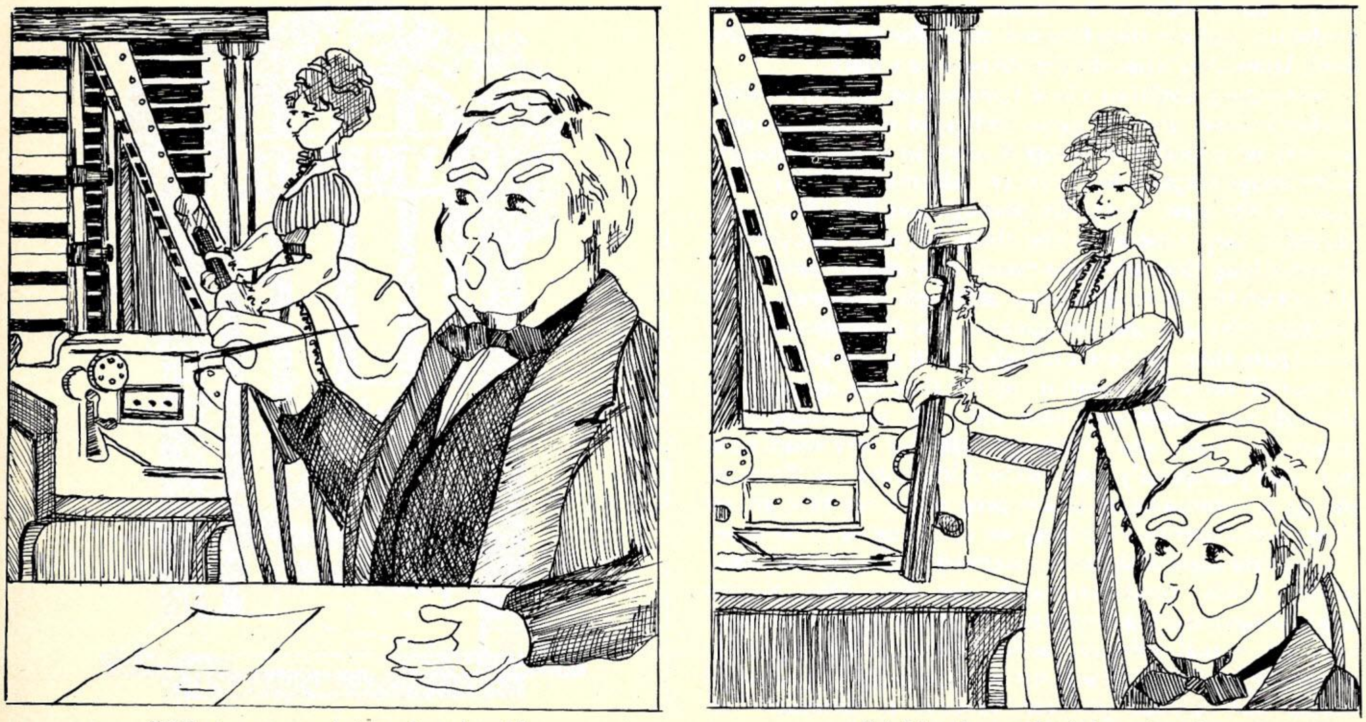
* * *

Personal computing "strictly confidential and subject to army censorship?" Better believe it's today, not fifty years from now. A handful of American university graduates in Israel helped put together some microcomputing courses and software for an institution of higher learning there. Sounded interesting, so ROM set off in search of the facts. Back came the report that not only weren't we cleared to ask them any questions, we weren't even allowed to find out who they are. Is the CIA watching your MPU?

Keep your I/ open,

ROMulus

BABBAGE AND LOVELACE



"What are you doing, Lovelace?"

"A bit of console debugging."

Cartoon by Linda Smythe

futuROMa

BIOFEEDFORWARD

by Bill Etra

First, I'm going to start out with a personal bias and premise. The personal bias is that I don't believe anything is impossible and along with that goes a feeling that if you believe things are impossible, you can't do them. But if you believe they *are* possible, you can do anything. You may never get around to it. You may not be able to do some things, but I really don't believe in limits to human endeavor.

I also believe most barriers are conceptual barriers. Roger Bannister broke the four-minute mile, by thinking about what happens when you can't breathe any more. The answer was nothing. So he just kept running until he couldn't breathe any more and then ran some more. He broke the fabled four-minute mile and has since been followed by many others.

Mixed in with this somewhere is part of Toynbee's theory of history which, badly mangled, states that great developments happen only on the marshlands of any one field. That they happen on the marshlands is precisely because people like Pasteur as a chemist don't know the rules of medicine, and are therefore not encumbered by them and don't know that what they're doing won't work.

Given this great blast a la *A Connecticut Yankee in King Arthur's Court* and the great feeling of moving forward, the way we conceive of things is open to a lot of change. Take image displays for instance. All image display takes place in the past. Certainly photographs are always displayed a long time after the thing happens. Movies are shown a long time after the events they record have actually occurred. As to television, although it appears to be real time it's really only pseudo-real time since transmission is no faster than the speed of light. Even if you are looking at yourself on a TV monitor, by the time the images pass through the cables and back out, you are seeing your image as it was a short time ago, albeit a very short time ago. One particularly noteworthy thing is that, as the technology of communications has progressed, we have gotten closer and closer to something we can truly call real time.

Given the state of the art in memory and computer technology today combined with the fact that signals originating in the brain are sent out to the muscles well before the actual physical action occurs—the electronic signal arrives at the muscle up to three-tenths of a second before the actual physical motion is initiated—an interesting body movement study offers itself.

There are some fairly good ideas about how you recognize patterns of overall body movements from specific points. For instance, the hip movement gives us a good idea of where the legs are going. The upper torso and the shoulder movement gives us a good idea of the arm movements, etc. Considering this, and using television technology, we can take a picture of where somebody is. This picture is only a very fraction of a second, say a thirtieth of a second, behind the person's actual movement, when we display it on a computer screen. If we input the information through a computer by way of a wire harness which gives us a pattern recognition from specific muscle points on the body and if we modify this information in the computer at very high speed, assuming a very high speed processor, it is theoretically possible to display a modified picture of where you are about to be. This would require probably an emitter-coupled logic processor using perhaps array processing, or possibly even matrix processing. But it could be done.

Now I am not saying this will display where you are about to be hours from now. Your image would only be, say about one-tenth of a second ahead of where you really are in the present. Perhaps the image would even be a little less than one-tenth of a second ahead of you. But ahead of you it would be.

This brings us to an interesting problem. This is actually a display of a first person personal future, or personal future tense, or personal future imperative tense. No matter what you do, no matter how you try to wiggle out of it, the display of the image on the monitor will always be a little ahead of where you are. Barring earthquakes which can be allowed for, etc., and outside forces changing the entire setup, you can't get away from the fact that you'll be displaying your personal future.

What we get into then is biofeedforward loops. Nobody

TICKETS
Please Note
As of today,
tomorrow's
train leaves
yesterday
. . .

Illustrated by Linda Smythe

knows what happens in the biofeedforward loop or how you adjust to the biofeedforward loops. We haven't even really finished exploring biofeedback loops and how one adjusts to them yet.

What really interests me is not that you can photograph a tenth of a second in the future so much as that by virtue of extending this pattern recognition into a whole personal history within the computer, you can theoretically begin to push it more and more into the future. Our concept of time and of data as always being in the past begins to change. On a personal level this looks like a nice area for us to explore in terms of the future of image display. When we do, we had better be prepared for a real future shock for there is no telling what will happen.

In *Foundation* Trilogy, Isaac Asimov proposed psychohistory, which is a high form of computer mathematics which allows you to tell what the whole future world will be, given the variables of biological mutation which are unknown or uncalculated. The whole *Foundation* Trilogy is about it, so the whole idea is not new. What is new is that now we are actually at the stage where we can begin to implement it, at least on an early experimental level. Certainly Xerox, IBM, and Bell Labs have the facilities for implementing it and I am going to try to talk with friends at Livermore Radiation Labs about trying to work out some elementary systems. I have already talked to IBM about testing the idea.

Actually, I suspect it has already been done in some cases. There are certain experiments where eye movement is studied by moving a dot on an oscilloscope screen according to electromyogram which is the muscle pulse of the eye. Some of these results have been rather surprising. In the cases producing unexpected results, it is possible that the projecting dot was slightly ahead of where the eye was moving because the experimenters were taking the controlling impulse from the muscle instead of from the actual physical eye movement.

As it turns out, the people have probably done it on a rudimentary level in the laboratory and just not realized that, on a fractional second basis, they were in a feedforward as opposed to a feedback mode.

This whole thing, by the way, is not unknown in Eastern cultures. There's the story of the Japanese martial-arts master who is in a room where they are trying to shoot him with wax bullets to test his abilities. Several people are shooting at him with these wax bullets and they just can't hit him.

They ask, "Why can't we hit you? Nobody is quick enough to dodge a bullet."

And the master, who I think was in his seventies at the time, answers, "I don't dodge your bullets. I dodge your intent to shoot."

Which explains how most oriental Zen or Taoist masters of martial arts think, namely that the key part of training is basically to learn to anticipate the next move of your opponent. This is really a personal method of telling where the opponent will be in the future before he gets there. Your ability to do this is practised over thirty years or so of initiation into the oriental martial arts. Given in that light, the concept becomes explainable or understandable. The point is that now we are about to do it on a much grander scale with computers. We are about to start backing into the future. Whether we like it or not. ▼

COMING UP IN *ROM*

- A Solar Eclipse Program
- An Inside Look at Crystal Growing
- Computer Pointillism
- Science and the Compulsive Programmer
- Medical Systems
- The Wrestling Program
- Flow Charts for Beginners
- Computers and Helen Keller
- Programming the School Lunch Program
- Building Your Graphics System

DON'T FORGET THE DIGITAL FOAM CONTEST

Beside our regular payment to contributors, we will be giving away two prizes for the best applications article on digital foam we receive before January 1, 1978.

First prize is your choice of $500 or its equivalent in Dynacon. Second prize is $250 or its equivalent in Dynacon.

The articles should be 1,500-3,000 words long. They should include diagrams (roughs are fine) and photographs of your system in operation. You must be able to demonstrate that you have it up and running.

For further details see *ROM,* Vol.I, No.1.

Dynacon material kits for experimentation and implementation are available from:

Dynacon Industries Inc.
14 Bisset Drive
West Milford, N.J. 07480

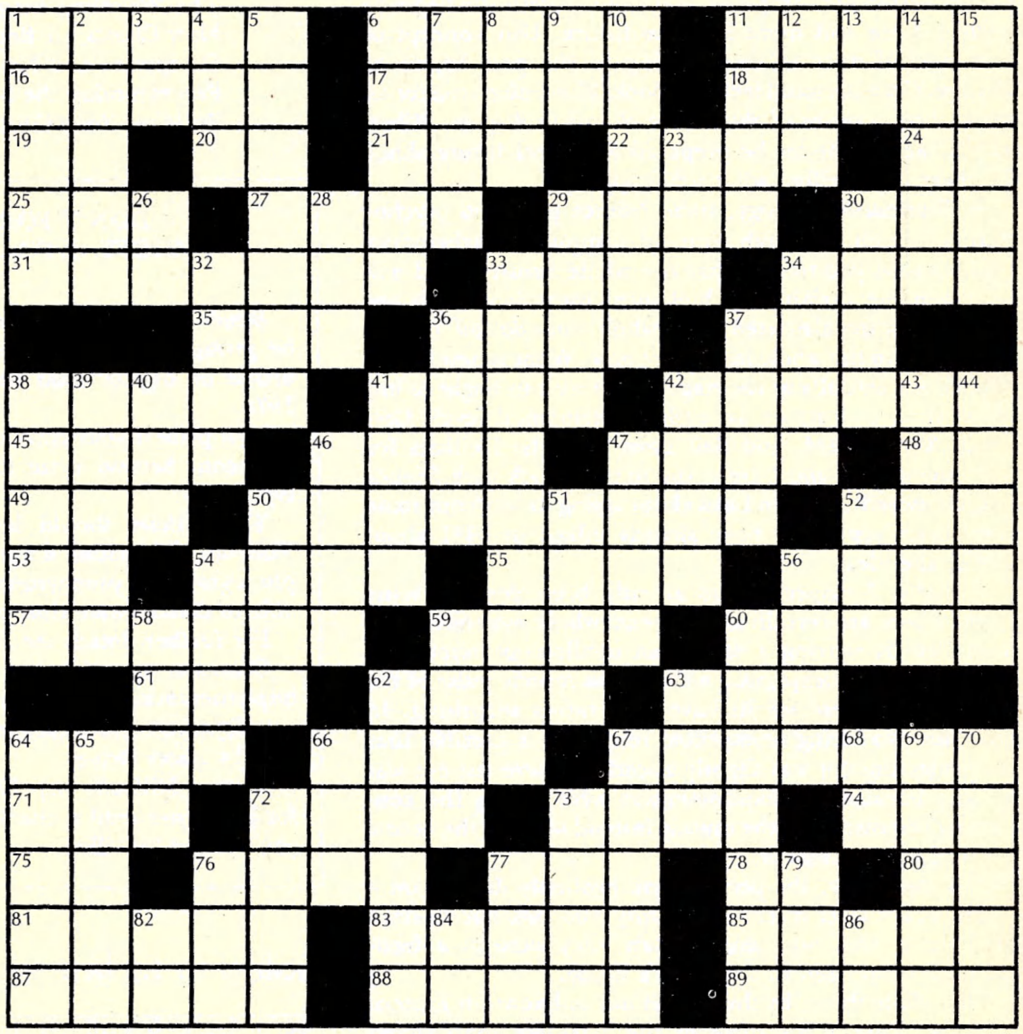
for $10 if you send a check with your order, or $12 if you wish to be billed.

Solution to last month's PROMpuzzle

S	L	A	M	■	S	H	O	E	■	■	A	P	U	P	■	■
L	I	M	A	■	Q	U	A	D	■	■	N	E	G	R	O	■
A	M	P	S	■	U	R	S	A	■	C	A	T	H	O	D	E
G	E	L	■	M	A	R	■	M	A	R	I	S	■	G	E	T
■	■	I	V	O	R	Y	■	■	R	E	S	■	E	R	S	E
A	C	T	I	V	E	■	R	E	T	E	■	C	R	A	S	S
C	O	U	P	E	■	R	U	N	S	■	K	A	R	M	A	■
E	N	D	■	■	S	A	L	T	■	L	A	M	E	■	■	■
R	E	E	F	■	T	H	E	R	M	A	L	■	D	A	S	T
■	■	■	A	M	Y	S	■	O	O	Z	E	■	■	U	P	I
■	R	A	D	I	X	■	A	P	S	E	■	S	A	T	A	N
G	E	N	E	S	■	T	O	Y	S	■	M	E	M	O	R	Y
A	M	A	S	■	T	I	N	■	■	L	O	G	I	C	■	■
M	O	L	■	C	O	M	E	T	■	U	S	O	■	O	M	A
S	T	O	R	A	G	E	■	S	U	R	F	■	E	D	I	T
■	E	G	Y	P	T	■	■	A	T	E	E	■	R	E	N	O
■	■	S	E	E	S	■	■	R	E	S	T	■	G	R	I	P

PROMpuzzle

by Daniel Alber



The solution to this PROMpuzzle will appear in next month's *ROM*

ACROSS

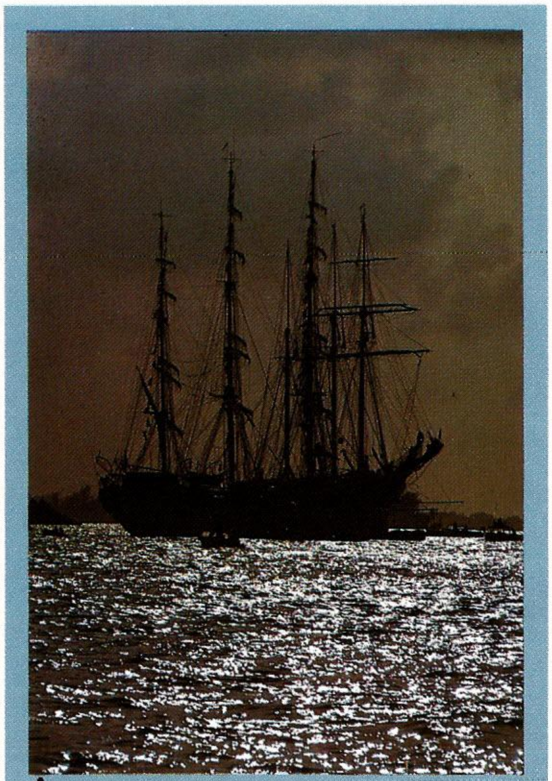
1. Singular input information
6. Silicon crystal device in computers
11. Unit in the MKSA system
16. Fat
17. Epic poem
18. Baked brick
19. Continent (abbr.)
20. Like
21. Harden
22. Prefix used with China
24. Capone or Capp
25. Operational training unit
27. Know in French (1st pers.)
29. To ____ (2 wds.)
30. Time zone (abbr.)
31. Checkpoint and____ procedures
33. Medicinal plant
34. Type of bargaining
35. Mature
36. USA for short
37. ____ air mail
38. DMA ____ stealing
41. Swiss mountains
42. ____ transducer
45. Hemp product
46. Man's name
47. Soaks up
48. Perform
49. Presidential nickname
50. Some memory
52. Political group (abbr.)
53. Myself
54. Gospel writer (abbr.)
55. Gad about
56. Born and____
57. ____ character
59. Decree
60. Approaches
61. Beast of burden
62. Menu
63. Singer Torme
64. Make coffee
66. Dock
67. System with little power dissipation
71. Unit of angular measurement (abbr.)
72. "What's My Line" alumnus
73. Overshoot
74. Animal foot
75. Operating system
76. Rational
77. "Peter ____"
78. Time interval for short
80. Six (Rom. num.)
81. Boolean operators
83. Real-time ____
85. Nostrils
87. Type of computer chip architecture
88. Storage devices
89. Board game

DOWN

1. Ion in a doped semiconductor crystal
2. Lessen
3. ____ Deum
4. Our nation for short
5. ____ control program
6. Numeral
7. French islands
8. Lubricate
9. Lawman (abbr.)
10. General purpose text program
11. Reductions in the received field strength
12. Fuss
13. Receive only
14. Humble
15. ____ clock
23. Born
26. You and me
28. Exist
29. Brews
30. Enthusiasm
32. Yarn
33. Device for controlling power
34. Swine
36. Landed
37. Big shots
38. Felony
39. Groups of heads in magnetic recording
40. Central processing element
41. Amo, amas, ____
42. Filth
43. Computer output device
44. Fills computer storage
46. Feminine suffix
47. Struck (archaic)
50. Drains
51. Communication line adapters for teletype
52. Constellation
54. Device used to shield portions of a base during a deposition process
56. Chime
58. Usually 7 3/8 by 3 1/4
59. Feudal estate
60. Feature of some data sets
62. Type of memory access
63. Metal-oxide semiconductor
64. ROMs with an added touch
65. Artist's stand
66. ____ and ink
67. Systems connecting two locations
68. Type of art
69. Rescues
70. Type of cheese
72. Example
73. Cudgel
76. Sensitivity time control
77. Soda
79. Cheer
82. East Indies (abbr.)
84. Musical syllable
86. Concerning

A magnificent first edition . . . pictorial memories of a moment America will never forget . . . or see again

THE SAILING SHIPS

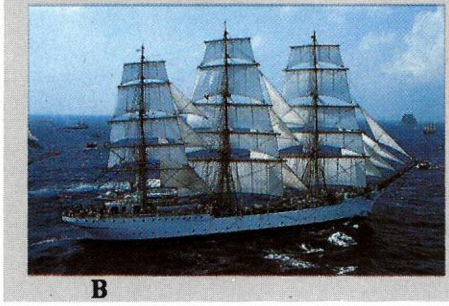


A

Professional quality lithographs of the majestic Bicentennial Ships, now available to the public in an exclusive, limited edition from America's foremost engraver-printer, COLLIER GRAPHICS.

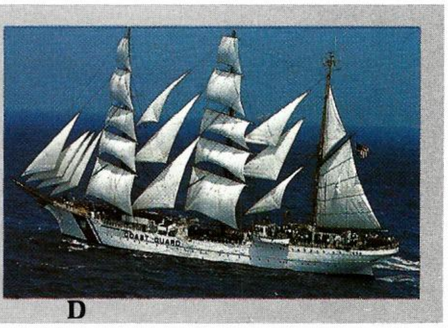
handsomely matted in silvery metal frames; $25 each

unframed, $10 each



B

C



D

THIS IS A LIMITED EDITION . . . ORDER TODAY WHILE THEY LAST

Never before offered to the public, these magnificent 12x19 full color lithographs of the tall-masted SAILING SHIPS were originally photographed for professional use only!

Now, you can own and enjoy their historic beauty, their incredible clarity, color and reproduction, which gives them almost a three-dimensional quality. And they are only available from COLLIER GRAPHICS — who provide the superb color graphics for America's leading advertising agencies and publishers.

Perfect for your home, boat or office. A lasting gift. Order a set for yourself and one for your children . . . to keep always.

COLLIER GRAPHICS INC., 240 West 40th Street, New York, New York 10018
Please rush the following SAILING SHIPS, securely packaged and postage prepaid, with the understanding that my purchase is UNCONDITIONALLY GUARANTEED by you if the order is returned within 15 days of delivery:

Send me the following print(s). Quantity ______ Price ______ Total ______

A. Christian Radich ______

B. Danmark ______

C. Kruzenshtern ______

D. Eagle ______

Please add $2.50 for shipping and handling for framed print(s) or $1.00 for unframed print(s).
(N.Y. State residents add applicable tax, NYC residents add 8%. Allow 6 weeks for delivery.)

Name ______

Address ______

City ______ State ______ Zip ______

Enclosed, check or money order for $______
Or charge my credit card ______ Master Charge ______ BankAmericard

Credit Card \# ______ Inter Bank \# ______ Expiration Date ______

Signature X ______

PROJECT
PROMETHEUS
needs you